Subject: Yet again, The Sky is Falling!
Posted by yp on Thu, 08 Mar 2007 18:12:12 GMT
View Forum Message <> Reply to Message

This is yet another floating point mystery, and I am unable to figure out which is the right way to deal:

I wrote this program ("Operation") in which I made sure that all calculations are done in double precision. The program accepts 6 mandatory arguments and returns the output to "result".

Syntax: Operation, A, B, C, D, E, F, result

I get variable results (after 3rd decimal point) when I pass some of the arguments as numbers and when I pass the same arguments as pre-defined variables.

Case#1:

```
IDL> Operation, A, B, 0.0D, 0.0D, 0.0D, F, result
IDL> help, result
RESULT          DOUBLE    = Array[7]

IDL> print, result
1.0247013       1.0279051       1.0365066       1.0447064
1.0477210       1.0543893       1.0569390
```

Case#2:

```
IDL> C = (D = (E = 0.0D))
IDL> Operation, A, B, C, D, E, F, result
IDL> help, result
RESULT          DOUBLE    = Array[7]

IDL> print, result
1.0250284       1.0281385       1.0367149       1.0450368
1.0480349       1.0547703       1.0573193
```

Why is such discrepancy? In my problem the accuracy after 3rd decimal point is not so important, however, after seeing the results I lose confidence on IDL's capability on Real number arithmetic!

May be I am missing something?

Thanks,
yas

Subject: Re: Yet again, The Sky is Falling!
Posted by yp on Fri, 09 Mar 2007 11:17:35 GMT
View Forum Message <> Reply to Message

Hi Carsten,

> what happens if you do
>
> int_LUT, 412.5, 1, 0.0D, 0.0D, 0.0D, 0.03D, result1
> int_LUT, 412.5, 1, null, null, null, 0.03D, result2
> int_LUT, 412.5, 1, 0.0D, 0.0D, 0.0D, 0.03D, result3
>
> is result1 eq result3?

Yes, result1 and result3 are identical.

> Is there any code in int_LUT that references files or
> COMMON blocks? Could that be called even though it is
> not supposed to be?

There is indeed a common block for the LUT, which is a table of pre-defined factors (in fact I added that before posting the code on the newsgroup, but the problem was still there even I read the LUT file inside int_LUT each time).

Subject: Re: Yet again, The Sky is Falling!
Posted by David Fanning on Fri, 09 Mar 2007 16:51:07 GMT
View Forum Message <> Reply to Message

yp writes:

> Your guess is right. I caught the culprit - I had presumed the first
> value of thetaV to be zero (which is not) and when I pass the
> parameter by reference it picks the nearest value of the thetaV tab
> (1.078) and it does exactly what you suspected .
>
> I did not realise that when I first wrote this last year... a classic
> example of reckless coding :(sorry to bother you all. But your
> suggestions were really helpful to figure this out :)

Ah, thanks. Just another reminder (as if we needed another) that MOST weirdness is self-inflicted. Good to know that some things are still right with the world. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Yet again, The Sky is Falling!
Posted by Michael Galloy on Fri, 09 Mar 2007 17:18:00 GMT
View Forum Message <> Reply to Message

On Mar 9, 9:38 am, "yp" <Yaswant.Prad...@gmail.com> wrote:
> Your guess is right. I caught the culprit - I had presumed the first
> value of thetaV to be zero (which is not) and when I pass the
> parameter by reference it picks the nearest value of the thetaV tab
> (1.078) and it does exactly what you suspected .

Ah, so the real problem was not passing by reference, but passing the same variable by reference to the routine multiple times and changing one of them (causing the others to also change).

I guess it makes sense, but somehow this was suprising to me. When I examined phi before and after the offending line:

```
print, phi
if (senzp LT senztab[0]) then senzp = senztab[0]
print, phi
```

the value of phi had changed.

Moral of the story: don't change the value of input parameters.

Mike
--
www.michaelgalloy.com

Subject: Re: Yet again, The Sky is Falling!
Posted by David Fanning on Fri, 09 Mar 2007 17:31:22 GMT
View Forum Message <> Reply to Message

mgalloy@gmail.com writes:

> Moral of the story: don't change the value of input parameters.

Except, of course, when that is the POINT of the input parameter.

Don't forget that. Otherwise, we could all use MatLab. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Yet again, The Sky is Falling!
Posted by Paul Van Delst[1] on Fri, 09 Mar 2007 18:27:21 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> mgalloy@gmail.com writes:
>
>> Moral of the story: don't change the value of input parameters.
>
> Except, of course, when that is the POINT of the input parameter.

But, then it's not an input parameter. It's an output.

I'm f95-centric, and I know it doesn't apply completely to IDL for a number of reasons, but I don't think people should new code where the functionality depends on *how* the arguments are passed (i.e. by value or reference).

It's a shame IDL doesn't allow for something similar to the fortran95 intent attribute for arguments, e.g.

```
function myfunc(x,y,z) result(a)
  real, intent(in) :: x, y, z
  real :: a
  ...
  x = 1.0  ! Illegal. Compiler will not allow mods to intent IN argument
  ...
end function myfunc
```

cheers,

paulv

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx

Subject: Re: Yet again, The Sky is Falling!
Posted by David Fanning on Fri, 09 Mar 2007 18:36:40 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:

> But, then it's not an input parameter. It's an output.

It generally has to get into the program somehow. :-)

> I'm f95-centric, and I know it doesn't apply completely to IDL for a number of reasons,
> but I don't think people should new code where the functionality depends on *how* the
> arguments are passed (i.e. by value or reference).

I'm not saying "pass by reference" is not dangerous, it is,
but it is no more dangerous, it seems to me, than a dynamically
(and weakly) typed language. Goodness, one of the reasons I LIKE
IDL is because you can do all these weird things that would
get your knickers in a snit in some other language.

Sure, you have to learn a few rules, and usually you learn
them the hard way, but you only have to learn them three or
four times before they get cemented in your brain. I think
it is a small price to pay for a LOT of power.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Yet again, The Sky is Falling!
Posted by Michael Galloy on Fri, 09 Mar 2007 19:13:55 GMT
View Forum Message <> Reply to Message

On Mar 9, 11:36 am, David Fanning <n...@dfanning.com> wrote:
> Paul van Delst writes:
>> But, then it's not an input parameter. It's an output.
>
> It generally has to get into the program somehow. :-)
>
>> I'm f95-centric, and I know it doesn't apply completely to IDL for a number of reasons,
>> but I don't think people should new code where the functionality depends on *how* the
>> arguments are passed (i.e. by value or reference).

>
> I'm not saying "pass by reference" is not dangerous, it is,
> but it is no more dangerous, it seems to me, than a dynamically
> (and weakly) typed language. Goodness, one of the reasons I LIKE
> IDL is because you can do all these weird things that would
> get your knickers in a snit in some other language.
>
> Sure, you have to learn a few rules, and usually you learn
> them the hard way, but you only have to learn them three or
> four times before they get cemented in your brain. I think
> it is a small price to pay for a LOT of power.

I agree that pass by reference can be dangerous and very useful. To clarify my "moral":

1. Document clearly which parameters are input, which are output, and which are both (i.e. modifying a variable "in place"). Don't change the input ones! If you need to have a default value for that input in the routine, create a new local variable.

2. Don't pass the same named variable as both an input and output parameter in a routine call. (Or as two output parameters!)

Mike
--
www.michaelgalloy.com

Subject: Re: Yet again, The Sky is Falling!
Posted by Paul Van Delst[1] on Fri, 09 Mar 2007 19:43:11 GMT
View Forum Message <> Reply to Message

mgalloy@gmail.com wrote:
> On Mar 9, 11:36 am, David Fanning <n...@dfanning.com> wrote:
>> Paul van Delst writes:
>>> But, then it's not an input parameter. It's an output.
>> It generally has to get into the program somehow. :-)
>>
>>> I'm f95-centric, and I know it doesn't apply completely to IDL for a number of reasons,
>>> but I don't think people should new code where the functionality depends on *how* the
>>> arguments are passed (i.e. by value or reference).
>> I'm not saying "pass by reference" is not dangerous, it is,
>> but it is no more dangerous, it seems to me, than a dynamically
>> (and weakly) typed language. Goodness, one of the reasons I LIKE
>> IDL is because you can do all these weird things that would
>> get your knickers in a snit in some other language.
>>
>> Sure, you have to learn a few rules, and usually you learn

>> them the hard way, but you only have to learn them three or
>> four times before they get cemented in your brain. I think
>> it is a small price to pay for a LOT of power.
>
> I agree that pass by reference can be dangerous and very useful. To
> clarify my "moral":
>
> 1. Document clearly which parameters are input, which are output, and
> which are both (i.e. modifying a variable "in place"). Don't change
> the input ones! If you need to have a default value for that input in
> the routine, create a new local variable.

You said it better than I did. What you said above is what I meant.

> 2. Don't pass the same named variable as both an input and output
> parameter in a routine call. (Or as two output parameters!)

Good lord. Why would anyone do *that*? (In any language) :o)

cheers,

paulv

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx

Subject: Re: Yet again, The Sky is Falling!
Posted by David Fanning on Fri, 09 Mar 2007 19:52:06 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:

>> 2. Don't pass the same named variable as both an input and output
>> parameter in a routine call. (Or as two output parameters!)
>
> Good lord. Why would anyone do *that*? (In any language) :o)

Well, because it seems "natural", for one thing. Consider
the POSITION keyword to TVIMAGE when you don't know for
sure if the user wants to preserve the aspect ratio of the
image or not, but you are keenly interested in knowing
where the image ended up.

That's, uh, about the only example I can thing of at the
moment, but it seems a good one. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Yet again, The Sky is Falling!
Posted by Paul Van Delst[1] on Fri, 09 Mar 2007 21:11:08 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Paul van Delst writes:
>
>>> 2. Don't pass the same named variable as both an input and output
>>> parameter in a routine call. (Or as two output parameters!)
>> Good lord. Why would anyone do *that*? (In any language) :o)
>
> Well, because it seems "natural", for one thing. Consider
> the POSITION keyword to TVIMAGE when you don't know for
> sure if the user wants to preserve the aspect ratio of the
> image or not, but you are keenly interested in knowing
> where the image ended up.

Well, of course. That's normal and I do that a lot (for the above purpose and also to prevent memory leaks) in f95 all the time. The syntax would be

```
real, intent(in out) :: position(:)
```

> That's, uh, about the only example I can thing of at the
> moment, but it seems a good one. :-)

That's not how I interpreted Mike's post. I meant if given a procedure like:

```
pro mypro, input_var, output_var
  output_var = input_var + 1.0
end
```

then don't call it thusly,

```
x = 1.0
mypro, x, x
```

I don't even know if it's possible in IDL... or f95..or C... or..? Hence my "why on

earth...?"

cheers,

paulv

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx

Subject: Re: Yet again, The Sky is Falling!
Posted by Michael Galloy on Fri, 09 Mar 2007 22:45:11 GMT
View Forum Message <> Reply to Message

On Mar 9, 2:11 pm, Paul van Delst <Paul.vanDe...@noaa.gov> wrote:
> That's not how I interpreted Mike's post. I meant if given a procedure like:
>
> pro mypro, input_var, output_var
> output_var = input_var + 1.0
> end
>
> then don't call it thusly,
>
> x = 1.0
> mypro, x, x
>
> I don't even know if it's possible in IDL... or f95..or C... or..? Hence my "why on
> earth...?"

Yes it's possible; that was what was happening in the code of the
original poster in this thread (he just wasn't thinking of his
"output_var" as an output). My suggestion is: don't do that.

Mike
--
www.michaelgalloy.com

Subject: Re: Yet again, The Sky is Falling!
Posted by Michael Galloy on Fri, 09 Mar 2007 22:53:07 GMT
View Forum Message <> Reply to Message

On Mar 9, 12:52 pm, David Fanning <n...@dfanning.com> wrote:
> Paul van Delst writes:
>>> 2. Don't pass the same named variable as both an input and output
>>> parameter in a routine call. (Or as two output parameters!)

>
>> Good lord. Why would anyone do *that*? (In any language) :o)
>
> Well, because it seems "natural", for one thing. Consider
> the POSITION keyword to TVIMAGE when you don't know for
> sure if the user wants to preserve the aspect ratio of the
> image or not, but you are keenly interested in knowing
> where the image ended up.
>
> That's, uh, about the only example I can thing of at the
> moment, but it seems a good one. :-)

I think you're thinking of a single parameter that is both an input
and an output. I think that is fine as long as that is clearly
documented. (READF/READU is another example of using this well.)

I'm talking about two parameters that are both passed a single named
variable (like Paul's mypro example). While legal, I don't think this
is a good technique for clear code.

Mike
--
www.michaelgalloy.com

Subject: Re: Yet again, The Sky is Falling!
Posted by James Kuyper on Sat, 10 Mar 2007 01:33:24 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:
...
> That's not how I interpreted Mike's post. I meant if given a procedure like:
>
> pro mypro, input_var, output_var
> output_var = input_var + 1.0
> end
>
> then don't call it thusly,
>
> x = 1.0
> mypro, x, x
>
>
> I don't even know if it's possible in IDL... or f95..or C... or..? Hence my "why on
> earth...?"

Well, it's trivially possible in every one of those languages - you've
just shown how to do it in IDL. It can even be a good idea, when the

function involved is documented as having been written to handle in-
place creation of an output array from an input array.

Subject: Re: Yet again, The Sky is Falling!
Posted by Sven Geier on Mon, 12 Mar 2007 22:04:10 GMT
View Forum Message <> Reply to Message

mgalloy@gmail.com wrote:

> I'm talking about two parameters that are both passed a single named
> variable (like Paul's mypro example). While legal, I don't think this
> is a good technique for clear code.
>

...and as usual there's exceptions where this is a perfectly good thing to
do. I have a routine in front of me that dynamically improves a "guess" of
some number. It takes an input and an output parameter and in almost all
cases you want to give it the same variable there. Schematically like this

```
 x = someOldGuess
 improve,x,x
```

where "improve" takes the first "x" as its input, copies the values to a
local variable, performs a bunch of magic and returns the result in
the "second x". From the outside, the variable "x" simply has a new,
improved value (which is the purpose of "improve'). However, there can be
extreme cases where one might want to have a little more introspection and
where one might want to check the returned value before using it and in
that case one can make it

```
 improve,x,y
 if (some test here) then x=y
```

or such.

I second David's statement that one of the nice things about IDL is that one
can do all these weird things. As someone once said (about C++, I
think) "all the power and all the elegance of a hand grenade": It ain't
always pretty but it gets things done.

- S

--
http://www.sgeier.net
My real email address does not contain any "Z"s.

Subject: Re: Yet again, The Sky is Falling!
Posted by Foldy Lajos on Mon, 12 Mar 2007 22:19:31 GMT
View Forum Message <> Reply to Message

On Mon, 12 Mar 2007, Sven Geier wrote:

> mgalloy@gmail.com wrote:
>
>> I'm talking about two parameters that are both passed a single named
>> variable (like Paul's mypro example). While legal, I don't think this
>> is a good technique for clear code.
>>
>
> ...and as usual there's exceptions where this is a perfectly good thing to
> do. I have a routine in front of me that dynamically improves a "guess" of
> some number. It takes an input and an output parameter and in almost all
> cases you want to give it the same variable there. Schematically like this
>
> x = someOldGuess
> improve,x,x
>
> where "improve" takes the first "x" as its input, copies the values to a
> local variable, performs a bunch of magic and returns the result in
> the "second x". From the outside, the variable "x" simply has a new,
> improved value (which is the purpose of "improve'). However, there can be
> extreme cases where one might want to have a little more introspection and
> where one might want to check the returned value before using it and in
> that case one can make it
>
> improve,x,y
> if (some test here) then x=y
>
> or such.
>
> I second David's statement that one of the nice things about IDL is that one
> can do all these weird things. As someone once said (about C++, I
> think) "all the power and all the elegance of a hand grenade": It ain't
> always pretty but it gets things done.
>
> - S
>
> --
> http://www.sgeier.net
> My real email address does not contain any "Z"s.
>

You can do that, but be cautios. Try this eg.:

```
pro improve, a, b
```

```
b=a+1
if a eq b then print, 'The Sky is Falling! (a is equal to a+1!)'
end

x=1
improve, x,x
```

I think it is better to use an improve function: x or y =improve(x).

regards,
lajos

Subject: Re: Yet again, The Sky is Falling!
Posted by Paul Van Delst[1] on Mon, 12 Mar 2007 22:30:11 GMT
View Forum Message <> Reply to Message

Sven Geier wrote:
> mgalloy@gmail.com wrote:
>
>> I'm talking about two parameters that are both passed a single named
>> variable (like Paul's mypro example). While legal, I don't think this
>> is a good technique for clear code.
>>
>
> ...and as usual there's exceptions where this is a perfectly good thing to
> do. I have a routine in front of me that dynamically improves a "guess" of
> some number. It takes an input and an output parameter and in almost all
> cases you want to give it the same variable there. Schematically like this
>
> x = someOldGuess
> improve,x,x
>
> where "improve" takes the first "x" as its input, copies the values to a
> local variable, performs a bunch of magic and returns the result in
> the "second x". From the outside, the variable "x" simply has a new,
> improved value (which is the purpose of "improve').

But why do that? It makes no difference since you're overwriting the original value of x anyway. Why not just do

```
x = someOldGuess
improve,x
```

where "improve" just modifies "x" internally as required. The only advantage (that I can see) of

improve,x,x

over

improve, x

is that the former serves to confuse the reader of the code. There may be examples of exceptions where the construct in question is a good idea, but this isn't one of them.

> I second David's statement that one of the nice things about IDL is that one
> can do all these weird things. As someone once said (about C++, I
> think) "all the power and all the elegance of a hand grenade": It ain't
> always pretty but it gets things done.

Yes, well, just because something *can* be done....

But I like the analogy - much more graphic than the old "shoot yerself in the foot" sawhorse. :o)

cheers,

paulv

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx

Subject: Re: Yet again, The Sky is Falling!
Posted by Michael Galloy on Mon, 12 Mar 2007 23:13:22 GMT
View Forum Message <> Reply to Message

Subject: Re: Yet again, The Sky is Falling!
Posted by JD Smith on Tue, 13 Mar 2007 00:18:40 GMT
View Forum Message <> Reply to Message

On Fri, 09 Mar 2007 11:13:55 -0800, mgalloy@gmail.com wrote:

> 2. Don't pass the same named variable as both an input and output
> parameter in a routine call. (Or as two output parameters!)

However, done carefully, this is a very useful method, but you're right that it can get you in trouble in a hurry.

JD

Subject: Re: Yet again, The Sky is Falling!
Posted by Paolo Grigis on Tue, 13 Mar 2007 10:24:16 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
> On Fri, 09 Mar 2007 10:25:41 +0100, Paolo Grigis wrote:
>
>> Here's a simple example of a routine that returns different values if it
>> is called by value or by reference.
>>
>> pro pg,a,b
>>
>> b=arg_present(a)
>>
>> end
>>
>>
>> IDL> x=1
>> IDL> pg,x,b
>> IDL> print,b
>> 1
>>
>> IDL> pg,x+0,b
>> IDL> print,b
>> 0
>>
>> IDL> pg,x[0],b
>> IDL> print,b
>> 0
>
> That's a bit of a contrived example, given that the whole reason I
> originally lobbied for ARG_PRESENT was so you could differentiate between
> by-value and by-reference arguments (which I observed IDL's own internal
> routines could do, so clearly it was functionality waiting to be exposed).
>
> JD
>

Sure, but it may nevertheless come handy for the next
submission to the obfuscated IDL code contest ;-)

Ciao,
Paolo