Subject: Re: Moving to objects loses array functionality?
Posted by Paul Van Delst[1] on Mon, 03 Nov 2008 23:18:18 GMT
View Forum Message <> Reply to Message

Demitri wrote:
> Hi,
>
> I'm trying to implement some of my data as proper IDL objects, but I'm
> either missing some syntax or else am losing some array functionality in
> the process.
>
> Let's say I have the following structures:
>
> void = {BOOK, $
> author : '', $
> pChapters : PTR_NEW()}
>
> void = {CHAPTER, $
> title : '', $
> pageCount : 0}
>
> pChapters is a pointer to contain a variable number of chapters in the
> BOOK structure. If I create an array of BOOKs, I can do this:
>
> IDL> print, books.author
>
> ...and get a list of all of the authors. I can also use utilities like
> WHERE of course.
>
> What I'm wondering is if I can extract a list of, say, all of the
> chapter names as a list (perhaps my example doesn't seem useful, but
> ignore that!). I was hoping something like this would work:
>
> IDL> print, (*books.pChapters).title
>
> ...especially since "books.pChapters" does return an array of pointers.
>
> Is there something I can do here without resorting to FOR loops within
> methods in the BOOK object?

The IDL documentation specifically states the dereference operator requires a scalar operand:

<quote>
Dereferencing Pointer Arrays

Note that the dereference operator requires a scalar pointer operand. This means that if you are dealing with a pointer array, you must specify which element to dereference. For example, create a three-element pointer array, allocating a new heap variable for each

element:

ptarr = PTRARR(3, /ALLOCATE_HEAP)

To initialize this array such that the heap variable pointed at by the first pointer contains the integer zero, the second the integer one, and the third the integer two, you would use the following statement:

FOR I = 0,2 DO *ptarr[I] = I
</quote>

So it would appear the facile answer to your question is: no. The documentation itself suggests the use of for-loops.

But, maybe some gun IDL guru out there may be able to conflate the histogram procedure and pointer dereferencing to do what you want? (I'd like to see that :o)

cheers,

paulv

Subject: Re: Moving to objects loses array functionality?
Posted by Anonymous on Tue, 04 Nov 2008 21:50:56 GMT
View Forum Message <> Reply to Message

Originally posted by: Demitri

Hi Paul,

On 2008-11-03 18:18:18 -0500, Paul van Delst <Paul.vanDelst@noaa.gov> said:

>
The IDL documentation specifically states the dereference operator
requires a scalar operand:
>
> <quote>
> Dereferencing Pointer Arrays
>
> Note that the dereference operator requires a scalar pointer operand.
> This means that if you are dealing with a pointer array, you must
> specify which element to dereference. For example, create a
> three-element pointer array, allocating a new heap variable for each
> element:
>
> ptarr = PTRARR(3, /ALLOCATE_HEAP)
>

> To initialize this array such that the heap variable pointed at by the
> first pointer contains the integer zero, the second the integer one,
> and the third the integer two, you would use the following statement:
>
> FOR I = 0,2 DO *ptarr[I] = I
> </quote>
>
>
> So it would appear the facile answer to your question is: no. The
> documentation itself suggests the use of for-loops.
>
> But, maybe some gun IDL guru out there may be able to conflate the
> histogram procedure and pointer dereferencing to do what you want?
> (I'd like to see that :o)
>
> cheers,
>
> paulv

Thanks for pointing that out - I missed that in the documentation. I can completely understand how assigning the values would require a FOR loop, but I was hoping that IDL could dynamically send the "message" to the (dereferenced) structure to read the value. I'll assume that this cannot be done until someone can prove me wrong!

Cheers,

Demitri
