Subject: Re: Initializing object array
Posted by btt on Tue, 09 Mar 2004 16:49:02 GMT
View Forum Message <> Reply to Message

Marc Schellens wrote:

```
> David Fanning wrote:
>
>> Dick Jackson writes:
>>
>>
>>>   class = { MYCLASS, contours:ObjArr(nElements)}
>>>
>>> would do it, but it will be a fixed number of elements, and the passed
>>> 'contours' would have to match that. If that's what you need, then fine,
>>> but I bet you need flexibility. All I can see for a solution right now
>>> is using a pointer:
>>>
>>>   class = { MYCLASS, contours:Ptr_New(/Allocate_Heap)}
>>>
>>> then, to assign it:
>>>   *self.contours = contours
>>>
>>> and to refer to one contour:
>>>   (*self.contours)[i]
>>>
>>> Sorry if I'm stating the obvious... or am *I* missing something?
>>
>>
>>
>> Well, after taking a nap I can see that it is going to
>> have to be a pointer, but I still can't see why. :-)
>>
>> I often use object containers to store objects, but I guess
>> this might have been the first time (at least in a while)
>> that I tried to store an object array. Oddly, an object
>> array is an object reference:
>>
>>    IDL> a = ObjArr(5)
>>    IDL> Help, a
>>         A     OBJREF    = Array[5]
>
>
> IDL> a=indgen(5)
> IDL> help,a
> A              INT       = Array[5]
>
> Its an *array* of object references, as 'a' is an *array* of INT.
```

```
>
>
>>  So you might think that if b was initialized as an object reference,
>>  you could store an object array in it. It should fit, it's just a
>>  long integer.
>>
>>    IDL> struct = {b:Obj_New()}
>>    IDL> struct.b = a
>>     % Expression must be a scalar in this context: A.
>>
>>  Of course, with a structure I can do this:
>>
>>    IDL> struct = {c:ObjArr(5)}
>>    IDL> struct.c = a
>>
>>  But I can't see a way to initialize an *object* like that. For example,
>>  this doesn't work:
>>
>>    FUNCTION MyProg::INIT, a
>>      self.c = ObjArr(5)
>>      self.c = a
>>      RETURN, 1
>>    END
>>
>>    PRO MyProg__Define
>>      class = {MYPROG, c:Obj_New()}
>>    END
>>
>>  When I run it, I get this:
>>
>>    IDL> d = Obj_New('myprog', a)
>>      % Expression must be a scalar in this context: <OBJREF Array[5]>.
>>
>>  Isn't that strange!?
>
>
>  Its all perfectly fine.
>  Maybe you used before a container?
>
```

Hello,

I agree with Marc.   The behaviour you are seeing is consistent with how
other variables behave. That's good, isn't it? It's written down that
it's good in one of these Coyote books sprawled on my desk.

I have a hunch you are going somewhere else with this, but would this do it?

```
   FUNCTION MyProg::INIT, a
     ok = self->IDL_CONTAINER()
     If ok Then $
 if n_elements(a) NE 0 then $
 For i = 0, n_elements(a)-1 Do self->Add, a[i]
     RETURN, ok
   END

   PRO MyProg__Define
     class = {MYPROG, INHERITS IDL_CONTAINER}
   END
```

Ben