Subject: Re: Dynamic array of an Object as class member variable.
Posted by David Fanning on Mon, 15 Aug 2005 13:02:57 GMT
View Forum Message <> Reply to Message

Chirag Modh writes:

> I want to implement dynamic array of an Object as class member
> variable.
>
> ;--
> My class
> Class= {myclass, $
> oLine: obj_new()$
> }
> Function myclass :: Init , N_element
> Self.oLine = Make_Array(N_element,/obj)
> For i=0 , N_element do begin
> Self.oLine[i] = obj_new('IDLgrPolyline')
> end
> End
>
> ;--
> I can't create dynamic array of an object as class member variable.
> Any other way, I can implement this thing.

I think if you returned a 1 from your INIT method you
would have better luck. :-)

Cheers,

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

Subject: Re: Dynamic array of an Object as class member variable.
Posted by Karl Schultz on Mon, 15 Aug 2005 17:01:48 GMT
View Forum Message <> Reply to Message

On Mon, 15 Aug 2005 07:02:57 -0600, David Fanning wrote:

> Chirag Modh writes:
>
>> I want to implement dynamic array of an Object as class member
>> variable.
>>

```
>>  ;-------------------------------------------------
>>  My class
>>  Class= {myclass, $
>>              oLine: obj_new()$
>>          }
>>  Function myclass :: Init , N_element
>>          Self.oLine = Make_Array(N_element,/obj )
>>          For i=0 , N_element do begin
>>        Self.oLine[i] = obj_new('IDLgrPolyline')
>>          end
>>  End
>>
>>  ;-----------------------------------------------------
>>  I can't create dynamic array of an object as class member variable.
>>  Any other way, I can implement this thing.
>
>  I think if you returned a 1 from your INIT method you
>  would have better luck. :-)
>
```

Yes, but there are other problems.

1) You can't have variable-sized things in a struct, but you can have pointers to variable-sized things in a struct. So the member variable needs to be a pointer.

2) The loop goes around one time too many.

```
Function myclass :: Init , N_element
       Self.oLine = PTR_NEW(Make_Array(N_element,/obj))
       For i=0 , N_element-1 do begin
     (*Self.oLine)[i] = obj_new('IDLgrPolyline')
       end
       return, 1
End

pro myclass__define
Class= {myclass, $
           oLine: ptr_new()$
        }

end
```

Of course the Cleanup method would need to call PTR_FREE to free oLine and you would need to think about when and how to destroy the polyline objects themselves.