Subject: Re: object creation question
Posted by davidf on Mon, 20 Apr 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Louis J. Wicker (l-wicker@tamu.edu) writes:

> I am trying to use objects to help develop a 2D plotting package for my
> cloud model data. I have run into a conceptual snag, and think that I am
> missing something obvious. Suppose I want to create an object which
> stores, amoung other things, a 2d array of data (like for imaging or
> contouring, etc). So I create the object using:
> --
> data2d = obj_new('scalar2d')
> --
> Then I define a structure definition as:
> --
> Pro scalar2d__define
>
> struct = {scalar2d, sdata: fltarr(nx,ny)}
>
> End
> --
>
> and then an init method:
>
> --
> Pro scalar2d::init
>
> array = fltarr(20,30)
>
> self.sdata = array
>
> End
> --
>
> But, of course, this fails, because I have to declare the SIZE of sdata in
> the procedure scalar2d__define. For various reason, I really dont want to
> have to know the size of the array, at least until the init method. I can
> pass the data in as part of the object creation, however, but I dont see
> how that helps, since the define procedure does not seem to recognize any
> arguments passed.
>
> I am new to this object stuff, and perhaps I am missing some information
> about named structure workings as well. I see that other object init
> methods in the object graphics library have this type of function (i.e.,
> IDLgrPlot creates an object which stores data passed in directly in the
> call, and it can be of various dimensions), yet I cant seem to figure out
> how to do this. Is there a trick, or am I missing something obvious?

You are absolutely right about not knowing the size of the data
at the time you create the object (and hence, the named structure
associated with the object). In general, if we need to create
a structure with a field that is undefined at the time the
structure is created, we use a pointer. And specifically,
we use a NULL pointer to hold the proper "space" in the
structure. (Similarly you use a null object if the field
will contain an object reference.)

For example, an "image" object might be defined like this:

```
PRO MYImage__Define
struct = { MYIMAGE, $
         image:Ptr_New(), $  ; Null pointer
         xsize:0, $
         ysize:0}
END
```

The INIT method would be defined like this:

```
FUNCTION MYImage::INIT, image
self.image = Ptr_New(image)
s = Size(*self.image)
self.xsize = s[1]
self.ysize = s[2]
RETURN, 1
END
```

Now, it is easy to write a new method, say the method REPLACE,
that replaces the original image with one of a different size:

```
PRO MYImage::REPLACE, newimage
*self.image = newimage
s = Size(*self.image)
self.xsize = s[1]
self.ysize = s[2]
END
```

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438

Subject: Re: object creation question
Posted by Evilio del Rio on Thu, 23 Apr 1998 07:00:00 GMT
View Forum Message <> Reply to Message

On Mon, 20 Apr 1998, Louis J. Wicker wrote:

> I am trying to use objects to help develop a 2D plotting package for my
> cloud model data. I have run into a conceptual snag, and think that I am
> missing something obvious. Suppose I want to create an object which
> stores, amoung other things, a 2d array of data (like for imaging or
> contouring, etc). So I create the object using:
>
> --
> data2d = obj_new('scalar2d')
> --
>
> Then I define a structure definition as:
>
> --
> Pro scalar2d__define
>
> struct = {scalar2d, sdata: fltarr(nx,ny)}
>
> End
> --
>
> and then an init method:
>
> --
> Pro scalar2d::init
>
> array = fltarr(20,30)
>
> self.sdata = array
>
> End
> --

Hi Louis,

You need to use pointers. That's the not so obvious bit you are missing.
Look the documentation for the procedures, PTR_NEW(), PTR_FREE,
PTR_VALID(), etc. Here's a bit of code that can do what you
want (more or less...):

```
FUNCTION scalar2d::init,DATA=data ; Init method must be a function.

 if (N_Elements(data) GT 1) then begin
  ; Put stuff to chek if input data is correct...
 endif else return,0

 self.sdata = PTR_NEW(data)
 return,1

End



Pro scalar2d::cleanup

 PTR_FREE,self.sdata ; Clean up allocated memory.

end



Pro scalar2d::plot

 plot, *self.sdata

end


Pro scalar2d__define

 struct = {scalar2d, sdata: PTR_NEW()}

End


---

data2d = obj_new('scalar2d',DATA=myData)

---


Hth.
Cheers,
```

Evilio Jose del Rio Silvan Institut de Ciencies del Mar
E-mail: edelrio@icm.csic.es URL: http://www.ieec.fcr.es/~evilio/

"Anywhere you choose,/ Anyway, you're gonna lose"- Mike Oldfield