Subject: Re: How to do nested objects??
Posted by Mark Hadfield on Tue, 24 Jun 2003 23:43:01 GMT
View Forum Message <> Reply to Message

"Jon Robinson" <wonjrobinson@erols.com> wrote in message
news:bdam9sati1@bob.news.rcn.net...
>

> I have a class that I want to have hold another object. My code
> compiles without error, however, when I try to run it, I get the
> error:
>
> % Variable is undefined: JWR_CAL_IMAGE_FILE_NAMES.
> % Execution halted at: JWR_CALIBRATIONNOTES__DEFINE 196
>
C:\RSI\IDL56\products\envi36\save_add\MultiSpectralImageCali bration__define.
> pro
> % OBJ_NEW
> % MULTISPECTRALCALIBRATION 13
> C:\RSI\IDL56\products\envi36\save_add\MultiSpectralCalibrati on.pro
> % $MAIN$
>
> The contained object is:
>
> PRO JWR_CAL_Image_File_Names__define
> struct = { JWR_CAL_Image_File_Names, InFileNames:STRARR(251) }
> END ; PRO JWR_CAL_Image_File_Names__define
>
> The containing object is:
>
> PRO JWR_CalibrationNotes__define
> struct = { JWR_CalibrationNotes, InFileName:'', $
> PathToDataFiles:'', $
> NumberOfNotes:0, $
> NotesArray:OBJARR(80), $
> CalibImageFileNames:JWR_CAL_Image_File_Names, ROIFileIDs:LONARR(4) }
> END ; PRO JWR_CalibrationNotes__define
>

> I get the error message quoted above when I try to create the
> containing object:

In the __define routine for the containing object, use obj_new()
instead of the name of the contained object.

```
PRO JWR_CalibrationNotes__define
struct = { JWR_CalibrationNotes, InFileName:'', $
PathToDataFiles:'', $
```

```
    NumberOfNotes:0, $
    NotesArray:OBJARR(80), $
    CalibImageFileNames:obj_new(), ROIFileIDs:LONARR(4) }
    END ; PRO JWR_CalibrationNotes__define
```

Then in the Init routine for the containing object, you must created the contained object and store a reference in the appropriate field of your class structure

```
    function JWR_CalibrationNotes::Init
    self.CalibImageFileNames = obj_new('JWR_CAL_Image_File_Names')
    ;; Do other initialisation stuff
    return, 1
    end
```

You'll almost certainly want to destroy the contained object in the Cleanup method also.

The thing to remember is that a __define procedure for a class structure (or for a named structure) only *defines* the structure. It does this by creating a prototype instance of the structure, then discarding it when the procedure terminates. The next time an object of the same class (or named structure of the same type) is created, it has the same fields and data types as the prototype, but none of the actual data.

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: How to do nested objects??
Posted by btt on Wed, 25 Jun 2003 12:38:49 GMT
View Forum Message <> Reply to Message

Mark Hadfield wrote:
> "Jon Robinson" <wonjrobinson@erols.com> wrote in message
> news:bdam9sati1@bob.news.rcn.net...
>
>
>> I have a class that I want to have hold another object. My code
>> compiles without error, however, when I try to run it, I get the
>> error:

<snip>

>
> In the __define routine for the containing object, use obj_new()
> instead of the name of the contained object.
>
>   PRO JWR_CalibrationNotes__define
>   struct = { JWR_CalibrationNotes, InFileName:'', $
>   PathToDataFiles:'', $
>   NumberOfNotes:0, $
>   NotesArray:OBJARR(80), $
>   CalibImageFileNames:obj_new(), ROIFileIDs:LONARR(4) }
>   END ; PRO JWR_CalibrationNotes__define
>
> Then in the Init routine for the containing object, you must created
> the contained object and store a reference in the appropriate field of
> your class structure
>
>   function JWR_CalibrationNotes::Init
>   self.CalibImageFileNames = obj_new('JWR_CAL_Image_File_Names')
>   ;; Do other initialisation stuff
>   return, 1
>   end
>
> You'll almost certainly want to destroy the contained object in the Cleanup
> method also.
>
> The thing to remember is that a __define procedure for a class
> structure (or for a named structure) only *defines* the structure. It
> does this by creating a prototype instance of the structure, then
> discarding it when the procedure terminates. The next time an object
> of the same class (or named structure of the same type) is created, it
> has the same fields and data types as the prototype, but none of the
> actual data.
>

Hi,

Just to add a bit to Mark's nice description. The values in the newly created and 'unpopulated' prototype are all set to 0 for numeric values, empty-string for strings, null-pointer for pointers and null-objects for objects.

Cheers,
Ben