Subject: Re: How to traverse/inquire a class object structure in IDL?
Posted by Paul van Delst on Wed, 13 Oct 1999 07:00:00 GMT
View Forum Message <> Reply to Message

First off, thanks to David and Pavel for their insights. Before I could check the newsgroup for replies, one of our younger go-getter science types came and told me something about object oriented programming that made good sense:

The data should be an attribute of the object, not the object itself.

Hmm. Anyway, he and I sat down for about 15 minutes and came up with the following class structure definition and cleanup method:

```
PRO nasti__define

; -- Define the NAMED data structure attribute
  data = { data, $
           wavenumber      : PTR_NEW(), $
           radiance        : PTR_NEW(), $
           altitude        : PTR_NEW(), $
           fov_angle       : PTR_NEW(), $
           fov_index       : PTR_NEW(), $
           latitude        : PTR_NEW(), $
           longitude       : PTR_NEW(), $
           aircraft_roll   : PTR_NEW(), $
           aircraft_pitch  : PTR_NEW(), $
           scan_line_index : PTR_NEW(), $
           date            : PTR_NEW(), $
           time            : PTR_NEW(), $
           decimal_time    : PTR_NEW() }

; -- Create object CLASS structure
  nasti = { nasti, $
            data : data }

END
```

I like this becuase now I can add additional attributes whenever I want, e.g. global attribute data read from the netCDF data file containing instrument calibration information and/or processing software CVS/RCS info etc.

The cleanup method is now:

```
PRO nasti::cleanup

  PRINT, FORMAT = '( /5x, "Clean up..." )'

; -- Free up pointers
```

```
  n_data_fields = N_TAGS( self.data )
  FOR i = 0, n_data_fields - 1 DO $
    IF ( PTR_VALID( self.data.(i) ) ) THEN $
      PTR_FREE, self.data.(i)

END
```

I just couldn't bring myself to typing PTR_FREE, self.whatever a bunch of times because if I ever change the data structure definition, I would have to change the cleanup as well. I like changes in my code to have as small a footprint as possible, i.e. change is required in as few places as possible. Dunno if that's a great idea but for my simple little example but it's a start. Right?

I wish I'd "discovered" objects earlier.....all that code I wrote that *needs* the data to be encapsulated. Crikey.

Thanks again!

paulv

--
Paul van Delst
Space Science and Engineering Center | Ph/Fax: (608) 265-5357, 262-5974
University of Wisconsin-Madison | Email: paul.vandelst@ssec.wisc.edu
1225 W. Dayton St., Madison WI 53706 | Web: http://airs2.ssec.wisc.edu/~paulv