Subject: Re: Need advice on building an object container
Posted by ben.bighair on Wed, 29 Apr 2009 11:58:23 GMT
View Forum Message <> Reply to Message

On Apr 27, 10:41 am, Paul van Delst <Paul.vanDe...@noaa.gov> wrote:
> Hello,
>
> I need a bit of advice on building a container for object.
>
> I have an object definition, let's say 'RTS', that contains the simulated
> top-of-atmosphere radiances for various satellites. I also have files that contains all
> the instance data for a whole bunch of those objects. Those data are organised by the
> sensor channel (i.e. different frequencies) and atmospheric profiles. So, in my
> pre-object-code I would create a structure array after determining the dimensions,
> something like
> Rts = PTRARR(n_Channels, n_Profiles)
> FOR m = 0L, n_Profiles-1L DO BEGIN
> FOR l = 0L, n_Channels-1L DO BEGIN
> Rts[l,m] = PTR_NEW({CRTM_RTSolution})
> ...read the current record...
> ENDFOR
> ENDFOR
>
> But now I want to use objects (mostly as a learning exercise, but also to stop folks from
> mucking about with the innards of the structure when they use it).
>
> So, the structure definition of RTS doubles as the object defn also. What do people
> recommend for reading the datafile? I see two options:
>
> 1) Use OBJARR(). Read dimensions and create the array and then fill it. However, this
> would still require the user to create the array after inquiring the file for its
> dimensions, and basically keep track of things.
>
> 2) Use a container. This is what I came up with this morning:
> PRO RTSfile__Define
> void = { RTSfile, $
> Filename : '', $
> FileId : 0L, $
> INHERITS IDL_Container }
> END
>
> and, in the RTSfile::Read method I would simply read each RTS object from the file and do a
> rtsfile->Add, rts
> But this method doesn't preserve the basic [n_Channels, n_Profiles] structure of the data.
> And, there's no indication in the RTSfile definition that this is a container for RTS
> objects, rather than a generic container -- but I'm wondering if worrying about that is
> just a distraction?
>

> 3) Sort-of combining (1) and (2) doing something like:
> PRO RTSfile__Define
> void = { RTSfile, $
> Filename : '', $
> FileId : 0L, $
> n_Channels : 0L, $
> n_Profiles : 0L, $
> rts : PTR_NEW() }
> END
> where, eventually,
> rts = PTR_NEW(OBJARR(n_Channels, n_Profiles))
> and then fill in the object references to the RTS object.
>
> So I was wondering how people would structure their code to handle this sort of thing. I
> think it's a common enough paradigm that a pattern probably exists but I haven't found it.
>
> Any info, hints, tips, appreciated.
>
> cheers,
>
> paulv

Hi Paul,

Have you considered crafting two new object containers - one for holding a vector of profile data-objects for a given channel and the other for holding an vector of channels. Assuming that the smallest reasonable element to craft an object around is one of your profile thingys, you could nest the containers something like this.

RTS is a container for
 RTS_ChannelsBucket is a container for
  RTS_ProfilesBucket is a container for
   RTS_ProfileElement

Your RTS object could have a method for retrieving a specific profile by channel and profile number...

myProfile = RTS::GetElement(thisChannel, thatProfile)

This method would then get thisChannel...

```
FUNCTION RTS::GetElement, thisChannel, thatProfile, count = count
  myChannel = self->Get(thisChannel, count = count)
  if (count GT 0) then begin
    profile = myChannel->Get(thatProfile, count = count)
  endif else profile = -1
  return, profile
```

end

Or perhaps method for retrieving all of the profiles for a given channel...

myChannel = RTS::GetChannel(thisChannel)

You probaby getthe idea. I suspect you could then manage all your file elements with pointers only at the "lowest" level, the profile elements. I don't know how much performance overhead this will add, but it should make it easier on you and will hide the darn elements from your users.

Cheers,
Ben
