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

Paul van Delst (paul.vandelst@ssec.wisc.edu) writes:

> 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.

Yeah, those young bucks know about object programming. :-(

> 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.

Not too bad, although I thought the first one was OK too.
Actually, it is the data and the methods that manipulate
the data that should be encapsulated in the object, so I

don't see that this new construction gains us much of anything,
except more structure de-referencing. :-)

But I *would* change the name of the structure from DATA
to something just a tad less generic. I see plenty of
trouble coming down the road with a name like DATA.

> 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

Have to admit that this is compact. :-)

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

You are on a slippery slope here. Once you fall
for objects almost *everything* looks like a perfect
opportunity to use one. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

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

---

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

> P.S. I think HEAP_GC is something you only use when
> you are alone in your office and the door is closed.
> I don't think it would inspire much confidence in your
> code to see it liberally sprinkled around everywhere.

Absolutely. However, it is quite useful on those late evenings when you are wishing there was a beer next to you and your Catch was commented out for some reason yesterday. Oh, those pointers - they never disappear unless you kill them, and you can't kill them if your program crashes... And here is the beautiful Heap_gc, what a marvel... Actually, if you ship out a compiled binary to your customer, they won't know...

Cheers,
Pavel

P.S. I don't use heap_gc in the code, only if my code crashes leaving dead pieces behind - then I call it from command line, making sure nobody is around watching me. Then I purge the log window to clean up the crimescene :-)

Subject: Re: How to traverse/inquire a class object structure in IDL?
Posted by davidf on Wed, 13 Oct 1999 07:00:00 GMT

Pavel Romashkin (promashkin@cmdl.noaa.gov) writes:

> Hey, I just got an idea! Why not do:
>
> obj_destroy, Nasti_instance
> heap_gc :-))) - that'll do the trick...

Sure, if your users don't mind waiting 30 seconds after
they quit your program to do something else. Perhaps you
could flash directions to the nearest coffee station
before you do the HEAP_GC. :-)

Cheers,

P.S. I think HEAP_GC is something you only use when
you are alone in your office and the door is closed.
I don't think it would inspire much confidence in your
code to see it liberally sprinkled around everywhere.

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: How to traverse/inquire a class object structure in IDL?
Posted by Pavel Romashkin on Wed, 13 Oct 1999 07:00:00 GMT

>> where the PtrHeapVar2 is the pointer to "self.wavenumber" and the object
>> reference is for the object. Cool.
>
> Cool if you have one object. Not so cool if you have several
> other programs with objects running, probably. :-)

Oh, tell me about it! I admit to having a bug (fixed now) in my widget program so severe that it crashed Xmanager and, of course, I could not clean up well. What do you do? - heap_gc. Poor program ran for almost 30 s finding more and more leftovers from my program and data, and overfilled my 200 line log window with all the findings. Amazingly enough, all that stuff was actually being cleaned up if the application quit gracefully.

Hey, I just got an idea! Why not do:

obj_destroy, Nasti_instance
heap_gc :-))) - that'll do the trick...

Cheers,
Pavel

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

Paul,
I see no flaw (at least on my programming level) in what you did. Why can't you just do the loop 13 times (this is the number of fields in your object structure, and it will never change since it is a named structure)? The method is tied to the type and 13 will always be OK, unless you are trying to write a generic method. If you want a generic method (that would wipe out pointers in any pointer-containing class), there is one way I saw it could be done. If you call "help, self, /object, /output=report" inside the cleanup method, then it returns also local information on the instance of that object, for example (I made a new method called "names"):

```
a -> names
** Object class NASTI, 0 direct superclasses, 1 known method
   Known Procedure Methods:
   NASTI::NAMES
   Instance Data:
     ** Structure NASTI, 13 tags, length=52:
     WAVENUMBER      POINTER   <NullPointer>
     RADIANCE        POINTER   <NullPointer>
     ALTITUDE        POINTER   <NullPointer>
     FOV_ANGLE       POINTER   <NullPointer>
     FOV_INDEX       POINTER   <NullPointer>
     LATITUDE        POINTER   <NullPointer>
     LONGITUDE       POINTER   <NullPointer>
     AIRCRAFT_ROLL   POINTER   <NullPointer>
     AIRCRAFT_PITCH  POINTER   <NullPointer>
     SCAN_LINE_INDEX POINTER   <NullPointer>
     DATE            POINTER   <NullPointer>
     TIME            POINTER   <NullPointer>
     DECIMAL_TIME    POINTER   <NullPointer>
```

Then you could programmatically inspect REPORT and find the number of tags in the object instance. Then, clean it up with a loop, checking for the type of the field being a pointer "if size(self.(i), /type) eq 10 then ptr_free, self.(i)".
I don't use objects of my own; so far I was happy with those IDL comes with. So, don't blame me if my approach is awkward. I just tried to come up with a solution. Maybe pros like David can give a much more elegant solution.

Good luck,
Pavel

> Not good. As more objects are created and destroyed, the valid pointer list
> grows. I would like to do the following in a CLEANUP method:
>
> FOR i = 0, n_object_structure_elements - 1 DO $
> IF (PTR_VALID(self.(i))) THEN $
> PTR_FREE, self.(i)
>
> that is, *explicitly* free up the pointers. This works great if I have a value
> for n_object_structure_elements.

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

Paul van Delst (paul.vandelst@ssec.wisc.edu) writes:

> Last night I entered into the world of IDL objects. I was amazed at how much
> easier it is to keep control of a data object rather than using a regular
> structure.

Hooray!

> Anyway, since I have been programming IDL in an Object Oriented mode for about 8
> hours, I have some questions that I hope someone out there can help me with. The
> documentation (on-line and printed) was not useful.

No, probably not. :-(

> I have a class structure definition in nasti__define.pro:.
> [Much deleted.]

> This all works fine. I have an simple inquire method:
>
> PRO nasti::inquire_nasti
>
> PRINT, FORMAT = '(/5x, "Inquiring...")'
> PRINT, PTR_VALID(), OBJ_VALID()
>
> END
>
> which when I run it, gives:
>
> IDL> n->inquire_nasti
> Inquiring...

> <PtrHeapVar2>
> <ObjHeapVar1(NASTI)>
>
> where the PtrHeapVar2 is the pointer to "self.wavenumber" and the object
> reference is for the object. Cool.

Cool if you have one object. Not so cool if you have several
other programs with objects running, probably. :-)

I think I would have written it something like this:

```
PRO nasti::inquire
  Print, 'Wave Number: ', *self.wavenumber
  Help, *.self.radiance, Output=thisOutput
  Print, 'Radiance Represented As: ', thisOutput
END
```

> Not good. As more objects are created and destroyed, the valid pointer list
> grows. I would like to do the following in a CLEANUP method:
>
> FOR i = 0, n_object_structure_elements - 1 DO $
> IF (PTR_VALID(self.(i))) THEN $
> PTR_FREE, self.(i)
>
> that is, *explicitly* free up the pointers. This works great if I have a value
> for n_object_structure_elements.
>
> QUESTIONS:
>
> 1) Is my technique valid? That is, I want to do the following:
> - create a data object
> - read some amount of data into that object
> - do stuff with the data object
> - delete the data object INCLUDING any pointers in the object.
> I don't know how much data I have ahead of time so I used pointers. Can I create
> data objects on the fly, based on how much data is in a datafile or requested
> from a datafile?
>
> 2a) If my technique is o.k., how do I free up the pointers in my object before I
> destroy it?

Your technique is probably OK, but it seems a bit
convoluted to me. Why not just write the CLEANUP routine
like this:

```
  PRO nasti::cleanup
  Ptr_Free, self.wavenumber
  Ptr_Free, self.radiance
```

```
   ...
   Ptr_Free, self.decimal_time
   END
```

A few extra keystrokes, perhaps, but it has the advantage
that you can see at a glance what it does. :-)

> ..OR..
>
> 2b) Is the above code stub a valid/smart way to free up the pointers in a data
> object and, if so, how do I determine the value of n_object_structure_elements?
> (You can't use N_TAGS() on an object but you can use the self.(i) type of
> structure reference so I'm confused.)

If you really like your solution, you could find
the number of fields in your object like this:

```
   thisClass = Obj_Class(self)
   ok = Execute("struct = {" + thisClass + "}")
   object_structure_elements = N_Elements(Tag_Names(struct))
```

But this just seems way too clever for me. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155