Subject: Re: Operator precedence
Posted by davidf on Mon, 10 Jul 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Harvey Rarback (rarback@ssrl.slac.stanford.edu) writes:

> I have a couple of questions regarding operator precedence. From this newsgroup
> and some experimentation I believe the following statement is true:
>
> Structure field extraction and array indexing have equal precedence, higher than
> pointer dereference but lower than parentheses to group expressions.
>
> Is this statement true?

True enough, I think. :-)

> So for nested structures struct1.struct2.data produces the same result as
> (struct1.struct2).data as expected. However, for nested objects (example code
> appended) these rules don't seem to apply:
>
> obj1.obj2.data produces an error
> (obj1.obj2).data produces the expected result, along with the infamous
> % Temporary variables are still checked out - cleaning up...
>
> Can some kind soul enlighten me about this behavior?

Oh, dear. :-(

I'm in the midst of a dozen things, Harvey, and I have
to teach courses the next couple of weeks. And JD is going
to give us the definitive answer anyway. But here is a quick
stab at this.

Your problem lies here:

> pro obj1__define
> obj1 = {obj1, obj2:obj_new()}
> end

Things would behave very much as you expect them to if
you had only *inherited* obj2 instead of putting it into
this structure as an object reference:

```
pro obj1__define
obj1 = {obj1, INHERTS obj2}
end
```

Now a statement like:

print, obj1.data

makes sense, since the data field is part of obj1 (via
the field that was inherited from obj2). But you
made a field in obj1 that is an object reference:

> obj1 = {obj1, obj2:obj_new()}

Hence, the only way to "see" that data field is to write
a method, since an object can only be dereferenced as a structure
in its own methods. That is why this statement:

Print, obj1.obj2.data

causes a problem. The obj2.data part is illegal. In fact, obj1.obj2 would
need to be a structure for the statement to be legal, and it is not.
It is an object. :-)

What would work is something like this (assuming you had written the
obj2__getdata method, of course):

Print, obj1.obj2->GetData()

> ; next line prints data but produces
> ; % Temporary variables are still checked out - cleaning up...
> print, (obj1.obj2).data

Yeah, I don't have a clue what this is doing, but anytime
you get that error message it sure as hell isn't what you
want to be doing. It's probably going crazy trying to
figure out what you had in mind. I think "Temporary variables
still checked out" is the IDL equivalent of throwing up your
hands and going to lunch in the real world. :-)

Hope this puts you on the right page, anyway.

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