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

David Fanning wrote:
>
> Harvey Rarback (rarback@ssrl.slac.stanford.edu) writes:
>
>> 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. :-)

Thanks for confirming this. Would someone who has clout with RSI get this tidbit into the documentation on Operator Precedence?

> 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:

I use object inheritance when it seems appropriate, but not in the instance I was complaining about. Let me explain. obj1 is a complicated widget application. obj2 is an object to replace widget_table. I would have logic and implementation issues with subclassing obj2:

logic: obj2 is not an "is a" to obj1. obj1 "has a" obj2.

implementation: making sure that the obj1 and obj2 have unique tags takes away some of the flexibility of having a standalone widget_table.

what if obj1 has multiple obj2's?

>> ; 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. :-)

I love your anthropomorphizing. It's what makes this newsgroup so much fun to read.

"J.D. Smith" wrote:
>
> You have found a bug in IDL's object data encapsulation code, which likely
> arises from the internal data-storage equivalence of structures and object
> heap data.

Ah, the predicted "definitive answer".

> I've found you can access the full data without error or warning if you use
> two steps:
>
> o=obj1.obj2
> print,o.data

I have discovered this too, but didn't want to let on in order to hear about other suggestions.

Thanks much to the two of you.

--Harvey
