Subject: Re: Still missing features in IDL 8
Posted by penteado on Thu, 04 Nov 2010 22:28:49 GMT
View Forum Message <> Reply to Message

On Nov 4, 4:29 pm, Bob <bobnnamt...@gmail.com> wrote:
> It seems to me that adding array subscripting to the list object is
> wrong since lists can take many more types of objects than arrays.
> This seems to be a product of IDL think that "everything is an array",
> which does not make sense anymore in 8.0. I think the proper way to
> implement the feature Paulo is asking for is to have an array_list
> object, which would only take arrays and could have overloaded
> brackets to get the extra subscripting. I think this is what Paulo is
> implementing already and it would seem it could be completely
> implemented with no changes to the language. Perhaps if you wanted to
> implement it in the language, an ARRAY keyword could be added to the
> list::init to force the list to only accept arrays and only if that is
> set then the bracket overloading could be added. It still seems
> cleaner to have a subclass object, however.

I disagree. I find bracket overloading to be one of the best points in IDL 8, and that brackets are proper for anything that may make use of them: arrays, lists, hashes, and whatever other classes, like those used by Graphics.

For that reason one important consideration in the class I am writing is to make sure that the other subscripts (those that come after the subscript for the list) will be passed on to whatever is in the list make use of them. I was initially thinking about only arrays in lists because that was the first need I encountered. But I realized that other subscriptable things should make use of that too. When that list class gets done and tested, I will probably extend that to a hash class, using some intermediate class to be inherited, which could be used for any other class that wanted that kind of forwarding for extra indices.

It is up to the user to know in what way the things being accessed can be subscripted, in the same way that the user must know, for instance, not to use more dimensions than an array has when subscripting it.