Subject: Re: Still missing features in IDL 8
Posted by penteado on Sat, 06 Nov 2010 21:51:23 GMT
View Forum Message <> Reply to Message

On Nov 6, 5:03 pm, JD Smith <jdtsmith.nos...@yahoo.com> wrote:
>> I agree that way would be better, not just because it would not mess
>> with the 8D limits. But that would require changes to the language,
>> for the new syntax, and a very different way for the overloadbrackets
>> method to work.
>
> I don't see how 8D limits come in. The limits are the number of
> possible arguments to the _OVERLOADBRACKETSLEFTSIDE method, which is
> much larger than 8.

The methods can take more arguments, but the parser currently only
allows 8 dimensions in the brackets. Which is why I said there are
currently two 8D limits: one for arrays, which is probably hard to
change, and one for the overloaded brackets, which can probably
increase easily (though at the cost of throwing errors if, say, 9D
were used on a class that had its methods written for only up to 8).

```
>  Unfortunately, the proposed method is deeply
> flawed, as there is a unavoidable ambiguity between array and list
> indices.  Consider:
>
> IDL> la=objarr(2,2)
> IDL> for i=1b,4 do la[i-1]=list([i,2*i^2],100b+[i,2*i^2])
> IDL> for i=0,3 do print,la[i]
>        1       2
>      101     102
>        2       8
>      102     108
>        3      18
>      103     118
>        4      32
>      104     132
>
> Now consider the proposed syntax: la[0,1,1]
>
> Which indices apply to which entity?  I.e., is this:
>
> IDL> print,(la[0,1])[1]
>      103     118
>
> or
>
> IDL> print,((la[0])[1])[1]
>      102
```

As IDL is now, la[0,1,1] cannot work. la is an array, not an object, so it will not take 3D indices as it is 2D. An error will be thrown, and no overload method will even be called. So parenthesis have to be used to select an individual element of la, which would be an object, and there is no problem.

>
> This also point out how the other syntax is superior:
>
> la[0,1][1] vs. la[0][1][1]
>
> would be different things. But I agree, this is not easy to
> accommodate at this point.

Yes, that would be better, but cannot be implemented without changing the language, parsers and interpreters, and only ITTVIS could do that.

>
> A few more comments on LIST and HASH:
>
> We need LISTARR and HASHARR to go with OBJARR.
>
> You should be able to assign a single scalar variable to "hash slices"
> to set each of the specified keys to that same value, just as you can
> assign a single value to a subset of lists:
>
```
> IDL> a=hash({one:1,two:2})
> IDL> a[ ['one','two'] ] = 4 ;; this should work!
> % Key and Value must have the same number of elements.
> % Error occurred at: HASH::_OVERLOADBRACKETSLEFTSIDE
> %                    $MAIN$
> % Execution halted at: $MAIN$
```

Yes, doing that should be possible (and I was surprised when I first noticed it is not), but no new data types (which is what a new array type would require) are needed. Only a simple change to the _overloadbracketsleftside methods.

Maybe I will add that to the methods I am writing. It would be implemented by either a foreach or a replicate. Probably a replicate, meaning a user would have to directly use a foreach instead when copying would be a problem, which seems far less likely to be the case: if the array of copies is big enough to be a problem, putting it into a list is probably a bigger, independent problem anyway, with the list overhead.