Subject: objarr
Posted by bjackel on Thu, 02 Sep 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Say I have a single object called Vector3, that allows nice things like

```
  mag= Vector3->magnitude()
  dot= Vector3->dotproduct(secondvector)
```

and so on. Then assume I've got a whole pile of vectors

```
  pile= OBJARR(100)
  FOR indx=0,99 DO pile[i]= Vector3
```

and want to manipulate them all at once

```
  magpile= pile->magnitude()
% Object reference must be scalar in this context: PILE
% Execution halted at:  $MAIN$
```

One foremost advantages of IDL is that logical groups of vector and matrix operations can be carried out with a single command. Am I correct in fearing that the only way to get what I want is with something like this?

```
  magpile= DBLARR(N_ELEMENTS(pile))
  FOR indx=0,N_ELEMENTS(pile)-1 DO magpile[indx]=
pile[indx]->magnitude()
```

Any comments or suggestions would be most welcome.

Brian Jackel

Subject: Re: objarr
Posted by Pavel Romashkin on Fri, 03 Sep 1999 07:00:00 GMT
View Forum Message <> Reply to Message

I agree that execution of a method on objarr of the same type would be very nice to have. For instance, I tend to have related arrays of the same type, let's say, IDLgrText (don't blame me for not having my own - built-in works for me so far, this is how inflexible I am :-( ). It'd be really handy to be able to do MyTextObjArr -> setProperty, color=[...] without going through a loop. I disagree that array operations should be disallowed for the sole reason being that an objarr can have different object types. I'd much rather have it to be the programmer's responsibility not to call a disallowed method on a heterogenious objarr than not having it at all :-(

Cheers,
Pavel

bjackel@phys.ucalgary.ca wrote:

> Say I have a single object called Vector3, that allows nice things like
>
> mag= Vector3->magnitude()
> dot= Vector3->dotproduct(secondvector)
>
> and so on. Then assume I've got a whole pile of vectors
>
> pile= OBJARR(100)
> FOR indx=0,99 DO pile[i]= Vector3
>
> and want to manipulate them all at once
>
> magpile= pile->magnitude()
> % Object reference must be scalar in this context: PILE
> % Execution halted at: $MAIN$
>
> One foremost advantages of IDL is that logical groups of
> vector and matrix operations can be carried out with a
> single command. Am I correct in fearing that the only
> way to get what I want is with something like this?
>
> magpile= DBLARR(N_ELEMENTS(pile))
> FOR indx=0,N_ELEMENTS(pile)-1 DO magpile[indx]=
> pile[indx]->magnitude()
>
> Any comments or suggestions would be most welcome.
>
> Brian Jackel

Subject: Re: objarr
Posted by steinhh on Fri, 03 Sep 1999 07:00:00 GMT
View Forum Message <> Reply to Message

In article <MPG.1238930f48dc31b9898da@news.frii.com>
davidf@dfanning.com (David Fanning) writes:
>
> bjackel@phys.ucalgary.ca (bjackel@phys.ucalgary.ca) writes:
>
>> Am I correct in fearing that the only
>> way to get what I want is with something like this?
>>
>> magpile= DBLARR(N_ELEMENTS(pile))

>>   FOR indx=0,N_ELEMENTS(pile)-1 DO magpile[indx]=
>> pile[indx]->magnitude()
>
> I'm afraid so. :-(

And I don't blame RSI for making it so. Just think about
it - an object array can be pointing to as many different
object types as the number of elements. There's no guarantee
that all of them have the particular method in question.
This goes for procedure methods as well as function methods.

Further, even if all the objects are of a single type,
a function method may return wildly different types of
results, depending on the internal state of the object!
So, there's no way for IDL to guess the type of array
needed to store the result.

OK, so there are ways to circumvent this, like determining
the type after all calls have been made. But, the scenario
also introduces a tricky problem with e.g. error reporting.
I guess you'd only get messages like

% Attempt to call undefined method: 'BLAH::MAGNITUDE()'

and various type conversion errors..

But I agree, it's annoyingly different from what we're used
to.

Regards,

Stein Vidar

Subject: Re: objarr
Posted by J.D. Smith on Tue, 07 Sep 1999 07:00:00 GMT
View Forum Message <> Reply to Message

bjackel@phys.ucalgary.ca wrote:
>
> Say I have a single object called Vector3, that allows nice things like
>
>   mag= Vector3->magnitude()
>   dot= Vector3->dotproduct(secondvector)
>
> and so on. Then assume I've got a whole pile of vectors
>
>   pile= OBJARR(100)

>   FOR indx=0,99 DO pile[i]= Vector3
>
> and want to manipulate them all at once
>
>   magpile= pile->magnitude()
> % Object reference must be scalar in this context: PILE
> % Execution halted at: $MAIN$
>
> One foremost advantages of IDL is that logical groups of
> vector and matrix operations can be carried out with a
> single command. Am I correct in fearing that the only
> way to get what I want is with something like this?
>
>   magpile= DBLARR(N_ELEMENTS(pile))
>   FOR indx=0,N_ELEMENTS(pile)-1 DO magpile[indx]=
> pile[indx]->magnitude()
>
> Any comments or suggestions would be most welcome.

Why not make your own array object as a container class for vector objects? You could hide all of this code behind an object interface, and also allow things like:

vec_pile->Rotate,!PI/2.

etc. And you could maintain the object list more consistently, ensuring it's integrity as Vector objects only.

While it might often be convenient for us to treat object arrays as traditional arrays, it's no more valid than expecting something like

a=[ptr_new(5),ptr_new('Foo')]
print,*a

to work. Objects have no single type or operational (binary, arithmetic, etc.) defaults, just as pointers have none. It's up to you to define these through the object oriented framework, which can be used to create much more elaborate and useful operant constructs (as in your example of dot products) than the traditional, static ones.

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-6263
304 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|