
Subject: Re: [Object IDL] routines that require user-supplied functions ...
Posted by [davidf](#) on Tue, 11 Aug 1998 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Darran Edmundson (dEdmundson@Bigfoot.com) writes:

- > Here is a *demo* object-IDL code I wrote to illustrate a problem.
- > This object integrates x^n over the interval (a,b) using the
- > QROMB function. QROMB requires a user-defined function but I
- > cannot manage to pass the 'poly' method.
- >
- > While this is a contrived example, one often wants to pass
- > object method functions/procedures to other IDL routines. Is
- > there a generic way of passing object methods to such intrinsic
- > routines?

I don't believe there is a generic way of passing object methods to intrinsic routines unless (perhaps) they are written in such a way as to expect them. It is clear QROMB has not been.

Nor have I found in the 15 minutes or so that I fooled around with this a completely satisfactory "object-like" way to solve this problem, although I did find a contrived solution that uses a common block.

The idea is this: Put the self object in a common block that can be declared in the POLY function. (It is a restraint of the QROMB routine that POLY be defined with one and only one parameter and that it be a vector of values for which the function is solved.) This gets the self object into the POLY function. BUT...not as a structure, as an object reference.

This means that the equation to be solved can't look like this, as it does in Darran's code:

```
x^(self.degree)
```

because the data of an object is hidden and is accessible only by its methods. So I had to write a Get_Degree method for the object that returns the degree of the function.

My solution looks like this:

```
function test4::init, degree, a, b
common polycommon, xx
; ~~~~~
```

```
function test4::integral
return, qromb('poly', self.a, self.b)
end
```

```
function poly, x
common polycommon, obj
;~~~~~
return, x^(obj->get_degree())
;~~~~~
end
```

```
pro test4__define
  struct = {test4, degree:0.0, a:0.0, b:0.0}
end
```

```
t = obj_new('test4',3.0,0.0,1.0)
```

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438, Toll Free Book Orders: 1-888-461-0155
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>