Subject: Object programming with data...
Posted by Randall Skelton on Sat, 18 May 2002 12:15:04 GMT
View Forum Message <> Reply to Message

Hi all,

I think I have found a bug in the IDL object programming interface which may help explain why there are no IDL object programming books. While IDL claims to support methods, there appears to be no mechanism for operator overloading!?!! Pardon my ignorance, but in a scientific programming language, what is the point supporting data encapsulation without operator overloading?

For those who have no idea what I am talking about, imagine I have a measurement that comprises two pieces of information. In this case the first variable is a string that describes the location of the measurement and the second variable is numeric vector or matrix of data. Surely I could store each piece of information separately but it makes a little more sense to store the two pieces of information in a structure. It is only a small leap to say that rather than having this be an IDL structure, it might be easier to make an IDL object. This way, I can benefit from the ability to define method routines that act directly on the data I have encapsulated.

```
a = obj_new('oop', 'test', randomu(seed,10))
b = obj_new('oop', 'test2', randomu(seed,10))

a->summary
OOP Type: test
OOP Data:
 0.624530     0.377744     0.539190     0.316347     0.976563     0.792964
 0.534616     0.138174     0.282342     0.907329
```

As a logical next step, I would like to be able to act directly on these objects with an operator. Perhaps, I would like to add them together. This seems straightforward enough, obviously it doesn't make sense to add the string variables but adding the data vectors seems straightforward.

```
IDL> print, a+b
% Operation illegal with object reference types.
% Execution halted at: $MAIN$
```

Surely there must be a way to add these variables and in most object oriented languages this is termed object overloading. I want to redefine the '+' sign so that IDL understands that when I write 'a + b' I really mean that I want to add the two data vectors (and possibly adjust the string to reflect that this is neither a nor b anymore). About the only way that you can do this in IDL is to define an 'add' function.

```
IDL> c = add(a,b)
IDL> c->summary
OOP Type: test+test2
OOP Data:
 1.45955      1.11524     0.702109      1.06542      1.13524      1.67752
 1.38169     0.799619     0.759985     0.953860
```

Sadly, life gets more complicated when you want to add more than two things together...

```
IDL> g = add(f,add(e,add(d,add(c,add(a,b)))))
```

Surely I must be missing some critical manual that describes the undocumented features of operator overloading. What I want to be able to write is:

```
IDL> g = a+b+c+d+e+f
```

We can talk about Methods, polymorphism, inheritance and persistence until we are all blue in the face, but without the ability to define operator methods the usefulness of IDL in programming with data is rather limited.

Any and all suggestions/comments are welcomed!

Cheers,
Randall

-- File: add.pro --

```
Function add, a, b, Type=type

  catch, theError
  if theError ne 0 then begin
    catch, /cancel
    message, !Error_State.Msg + ' Returning...', /info
    return, -1
  endif

  if n_params() lt 2 or size(a, /type) ne 11 or size(b, /type) ne 11 then $
    message, 'Must pass two objects to add', /noname

  if obj_isa(a,'oop') eq 0 or obj_isa(b,'oop') eq 0 then $
    message, 'Passed objects must be of class oop', /noname

  ; keywords
  if n_elements(type) ne 0 then begin
    if size(type, /type) ne 7 then message, 'Type must be a string', /noname
    ret_type = type
```

```
  endif else ret_type = a->GetType() + '+' + b->GetType()

  ret_data = a->GetData() + b->GetData()

  return, obj_new('oop', ret_type, ret_data)
End

-- File: oop__define.pro --

 ;--------------------------------------------------------- --------------
Function OOP::GetType
  return, self.type
End

 ;--------------------------------------------------------- --------------
Function OOP::GetData
  return, *self.data
End

 ;--------------------------------------------------------- --------------
Pro OOP::Summary, _Extra=extra

  print, 'OOP Type: ', self.type
  print, 'OOP Data: '
  print, *self.data, _Extra=extra

End

 ;--------------------------------------------------------- --------------
Pro OOP::Cleanup

  ptr_free, self.data

End

 ;--------------------------------------------------------- --------------
Function OOP::Init, type, data

  catch, theError
  if theError ne 0 then begin
    catch, /cancel
    message, !Error_State.Msg + ' Returning...', /info
    return, -1
  endif

  if n_elements(type) eq 0 then $
    message, 'Must give a type label', /noname
```

```
  if size(type, /type) ne 7 then $
    message, 'Type must be a string', /noname

  if n_elements(data) eq 0 then $
    message, 'Must give some data', /noname

  if size(data, /n_dimensions) gt 2 then $
    message, 'Data must be either a vector or a matrix'

  idx = where([2,3,4,5,6,9] eq size(data, /type))
  if idx[0] eq -1 then message, 'Data must be numeric'

  self.type = type
  self.data = ptr_new(data)

  return, 1

End

 ;--------------------------------------------------------- ----------
Pro oop__define

  s = { oop, $
        type:   '', $
        data:   ptr_new() $
      }

End
```

Subject: Re: Object programming with data...
Posted by David Fanning on Thu, 23 May 2002 16:58:44 GMT
View Forum Message <> Reply to Message

Pavel A. Romashkin (pavel_romashkin@hotmail.com) writes:

> One thing I *am*
> disappointed with is that David apparently lost interest in microbrews
> and I have not found another lure for him yet.

One thing *I* am disappointed in is that when you
get to be my age brewskis and fast women are
relegate to memory only. :-(

Cheers,

David

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

Subject: Re: Object programming with data...
Posted by David Fanning on Thu, 23 May 2002 17:04:47 GMT
View Forum Message <> Reply to Message

David Fanning (david@dfanning.com) writes:

> One thing *I* am disappointed in is that when you
> get to be my age brewskis and fast women are
> relegate to memory only. :-(

Uh, that should be "relegated". Damn, that syntax
of Pavel's is a siren's call. :-(

Cheers,

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