

---

Subject: syntax for calling parent class `_overloadPlus` method  
Posted by [Markus Schmassmann](#) on Thu, 28 Apr 2016 13:01:03 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

i'm trying to overload operators for my subclass of `idl_variable`, but don't find the correct syntax for calling the parent classes' operator function.

How do i have to correct the line below marked ';problem' without using 'left+right'?  
i guess i have to put something like `XXX.idl_variable::_overload...`  
but what would then be XXX?

```
---
pro sandbox__define
  struct={sandbox, $
    inherits idl_variable, $
    reps: ptr_new() $
  }
end

function sandbox::Init, array, reps
; a bit of code
void=self.idl_variable::init()
void=self.idl_variable::set_value(array)
*self.reps=reps
return, 1
end

function sandbox::_overloadPlus, left, right
; some code
out=idl_variable::_overloadPlus(left,right) ;problem
; some more code
return, out
end
---
```

PS: Sorry, stupid question of a beginner, but i failed to find the solution elsewhere.

PPS: There may be more errors, but the rest at least compiles.

---

---

Subject: Re: syntax for calling parent class `_overloadPlus` method  
Posted by [Michael Galloy](#) on Thu, 28 Apr 2016 20:35:40 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On 4/28/16 7:01 AM, Markus Schmassmann wrote:  
> i'm trying to overload operators for my subclass of `idl_variable`, but

```

> don't find the correct syntax for calling the parent classes' operator
> function.
>
> How do i have to correct the line below marked ';problem' without using
> 'left+right'?
> i guess i have to put something like   XXX.idl_variable::_overload...
> but what would then be XXX?
>
> ---
> pro sandbox__define
>   struct={sandbox, $
>     inherits idl_variable, $
>     reps:   ptr_new() $
>   }
> end
>
> function sandbox::Init, array, reps
>   ; a bit of code
>   void=self.idl_variable::init()
>   void=self.idl_variable::set_value(array)
>   *self.reps=reps
>   return, 1
> end
>
> function sandbox::_overloadPlus, left, right
>   ; some code
>   out=idl_variable::_overloadPlus(left,right)   ;problem
>   ; some more code
>   return, out
> end
> ---
>
> PS: Sorry, stupid question of a beginner, but i failed to find the
> solution elsewhere.
> PPS: There may be more errors, but the rest at least compiles.

```

In general, you would use something like the following to call a parent's implementation:

```
out = self->IDL_Variable::overloadPlus(left, right)
```

You can use the . notation you used in ::init as well:

```
out = self.IDL_Variable::overloadPlus(left, right)
```

But, in your case, you are calling some methods that don't exist. As far as I can tell, there are no IDL\_Variable::init, IDL\_Variable::set\_value, and IDL\_Variable::\_overloadPlus methods.

Mike

--

Michael Galloy

www.michaelgalloy.com

Modern IDL: A Guide to IDL Programming (<http://modernidl.idldev.com>)

---

---

Subject: Re: syntax for calling parent class `_overloadPlus` method  
Posted by [Markus Schmassmann](#) on Fri, 29 Apr 2016 09:45:09 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On 04/28/2016 10:35 PM, Michael Galloy wrote:

> On 4/28/16 7:01 AM, Markus Schmassmann wrote:

>> i'm trying to overload operators for my subclass of `idl_variable`, but  
>> don't find the correct syntax for calling the parent classes' operator  
>> function.

>>

>> How do i have to correct the line below marked ';problem' without using

>> 'left+right'?

>> i guess i have to put something like `XXX.idl_variable::_overload...`

>> but what would then be XXX?

>>

>> ---

>> pro sandbox\_\_define

>> struct={sandbox, \$

>> inherits idl\_variable, \$

>> reps: ptr\_new() \$

>> }

>> end

>>

>> function sandbox::init, array, reps

>> ; a bit of code

>> void=self.idl\_variable::init()

>> void=self.idl\_variable::set\_value(array)

>> \*self.reps=reps

>> return, 1

>> end

>>

>> function sandbox::\_overloadPlus, left, right

>> ; some code

>> out=idl\_variable::\_overloadPlus(left,right) ;problem

>> ; some more code

>> return, out

>> end

>> ---

>>

>> PS: Sorry, stupid question of a beginner, but i failed to find the

>> solution elsewhere.  
>> PPS: There may be more errors, but the rest at least compiles.  
>  
> In general, you would use something like the following to call a  
> parent's implementation:  
>  
> out = self->IDL\_Variable::overloadPlus(left, right)  
>  
> You can use the . notation you used in ::init as well:  
>  
> out = self.IDL\_Variable::overloadPlus(left, right)  
>  
> But, in your case, you are calling some methods that don't exist. As  
> far as I can tell, there are no IDL\_Variable::init,  
> IDL\_Variable::set\_value, and IDL\_Variable::\_overloadPlus methods.  
>  
> Mike  
when searching for the definition of the variable class i found a file  
idl\_variable\_\_define.pro in the following path:  
/opt/idl/idl\_local/pub\_domain/ssw/gen/idl/clients/rpc/  
(i don't maintain that file tree)  
it does have an ::init & ::setvalue function.

As for the \_overloadPlus, i may apparently erred in assuming  
[http://www.harrisgeospatial.com/docs/Overloadable\\_Operators.html](http://www.harrisgeospatial.com/docs/Overloadable_Operators.html)  
applies also to variables, i assumed them not being mentioned in the  
\_\_define file meant they were implemented in C but still could be  
accessed as i wanted.

Then let me rephrase the question:

What class(es) should i use as parent class, if i want to create a  
class, that during initialisation or initial set\_value accepts an array  
and a repetition pattern, and afterwards should behave as if the  
sandboxMember has been expanded from the array & pattern using a  
combination of rebin,reform,transpose...  
sandbox(indgen(1,10),[40, 1,60]) should behave the same as  
rebin(indgen(1,10),[40,10,60]), but only use the memory of indgen(10)  
and an array along[8] and run faster. The operators themselves when not  
operating on trivial cases will have to be implemented in C.

i want to be able to pass my sandboxMembers to foreign code that should  
not realize it has not been passed an ordinary array. It will not be a  
small thing to do, but if done right should increase idl performance  
quite a bit.

PS: sorry for the double-post before

---

---

Subject: Re: syntax for calling parent class `_overloadPlus` method  
Posted by [Michael Galloy](#) on Tue, 03 May 2016 17:05:53 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On 4/29/16 3:45 AM, Markus Schmassmann wrote:

- > when searching for the definition of the variable class `i` found a file
- > `idl_variable__define.pro` in the following path:
- > `/opt/idl/idl_local/pub_domain/ssw/gen/idl/clients/rpc/`
- > (i don't maintain that file tree)
- > it does have an `::init` & `::setvalue` function.

I actually have an `IDL_Object` class in my library as well. It is useful if I want the object to have overloaded operators when running in IDL 8.0+, but still compile and is usable (without the operators) when running on less than IDL 8.0.

- > As for the `_overloadPlus`, i may apparently erred in assuming
- > [http://www.harrisgeospatial.com/docs/Overloadable\\_Operators.html](http://www.harrisgeospatial.com/docs/Overloadable_Operators.html) applies
- > also to variables, i assumed them not being mentioned in the `__define`
- > file meant they were implemented in C but still could be accessed as i
- > wanted.

From the docs for `IDL_Object`:

- > This class serves as an interface class for other classes. There is never a need to instantiate an `IDL_Object` class directly.

What I understand this to mean, is that there is no implementation at all of `IDL_Object`, it is just a definition which marks an object as possibly having operator overloaded methods. I thought there was something more in the docs about this, but I couldn't find it.

- > Then let me rephrase the question:
- >
- > What class(es) should i use as parent class, if i want to create a
- > class, that during initialisation or initial `set_value` accepts an array
- > and a repetition pattern, and afterwards should behave as if the
- > `sandboxMember` has been expanded from the array & pattern using a
- > combination of `rebin`, `reform`, `transpose`...
- > `sandbox(indgen(1,10),[40, 1,60])` should behave the same as
- > `rebin(indgen(1,10),[40,10,60])`, but only use the memory of `indgen(10)`
- > and an array `ulong[8]` and run faster. The operators themselves when not
- > operating on trivial cases will have to be implemented in C.
- >
- > i want to be able to pass my `sandboxMembers` to foreign code that should
- > not realize it has not been passed an ordinary array. It will not be a
- > small thing to do, but if done right should increase idl performance
- > quite a bit.
- >

> PS: sorry for the double-post before

You will have to inherit from IDL\_Object to get operator overloading, but you will have to implement all code yourself, you will get nothing from IDL\_Object.

Mike

--

Michael Galloy

[www.michaelgalloy.com](http://www.michaelgalloy.com)

Modern IDL: A Guide to IDL Programming (<http://modernidl.idldev.com>)

---