Fortran classes and objects

Victor Eijkhout and Carrie Arnold and Charlie Dey

Fall 2017

Classes and objects

Fortran classes are based on `type` objects, a little like the analogy between C++ `struct` and `class` constructs.

New syntax for specifying methods.

Object is type with methods

You define a type as before, with its data members, but now the type has a contains.

```
Type,public :: Point
   real(8) :: x,y
 contains
   procedure, public :: setzero
   procedure, public :: set
   procedure, public :: length
   procedure, public :: distance
End type Point
```

Methods have object as argument

You define functions that accept the type as first argument, but instead of declaring the argument as type, you define it as class.

The members of the class object have to be accessed through the % operator.

```
subroutine set(p,xu,yu)
  implicit none
  class(point) :: p
  real(8),intent(in) :: xu,yu
  p%x = xu; p%y = yu
end subroutine set
```

TACC

Objects definition, method invocation

Class objects are defined as type objects, just as if there were no class functions on them. The class functions are accessed as

```
object%function(arg1,arg2)
```

where the arguments do not include the class argument.

```
use PointClass
implicit none
type(Point) :: p1,p2

call p1%set(1.d0,1.d0)
call p2%set(4.d0,5.d0)
```

Use modules!

It is of course best to put the type definition and method definitions in a module, so that you can use it.

Mark methods as `private` so that they can only be used as part of the `type`:

```
Module PointClass
  private
contains
  subroutine setzero(p)
    implicit none
    class(point) :: p
    p%x = 0.d0 ; p%y = 0.d0
  end subroutine setzero
End Module PointClass
```

Point program

```
Module PointClass                          end subroutine set
  Type,public :: Point
    real(8) :: x,y                    End Module PointClass
   contains
     procedure, public :: set       Program PointTest
     procedure, public :: distance  use PointClass
  End type Point                      implicit none
                                      type(Point) :: p1,p2
private
contains                             call p1%set(1.d0,1.d0)
                                     call p2%set(4.d0,5.d0)
  subroutine set(p,xu,yu)
    implicit none                    print *,"Distance:",p1%distance(p2)
    class(point) :: p
    real(8),intent(in) :: xu,yu    End Program PointTest
    p%x = xu; p%y = yu
```

TACC

Exercise 1

Take the point example program and add a distance function.