Chapter 5

Object Oriented Programming

5.1 Derived Types

They are a way to group connected variables. Where you declare variables you make up your own data_type

```
Derived Type ex1

TYPE COORDINATES
    REAL(KIND=dp) :: x, y, z
END TYPE COORDINATES

TYPE(COORDINATES) :: point1, point2

point1%x=1.0
point1%y=2.0
point1%z=3.0
write(*,*) point1

point2=coordinates(4.0, 5.0, 6.0)
write(*,*) point2
```

5.2 Modules

In general Modules have the form

```
General Module Structure

MODULE module_name
    static data definitions
CONTAINS
    functions, subroutines and interfaces
ENDMODULE module_name
```

double example

double.f90

```
MODULE double
    IMPLICIT NONE
    INTEGER, PARAMETER :: DP = SELECTED_REAL_KIND(P=15)
ENDMODULE double
```

Now in your main program you start
5.3 Interfaces

So now we know the major two features of modules. We can organize data and functions. Now let’s learn about interfaces which will make this module easier to use. So to do this we first need two functions. We’ll make two functions which assign values to the matrices.

```
TYPE Matrix
  REAL(KIND=DP) :: element
END TYPE Matrix
!
CONTAINS!
!
Real(kind=dp) FUNCTION OneNorm(mat)
  TYPE(Matrix), intent(in) :: mat(:,:)
  Real(kind=dp) :: colsum
  integer :: j, n, m
  m=size(mat(:,1))
  n=size(mat(1,:))
  OneNorm=0.0_dp
  do j=1,n
    colsum=sum(abs(mat(:,j)%element))
    OneNorm=Max(OneNorm, colsum)
  enddo
END FUNCTION OneNorm

subroutine matrix_real_scalar(mat_scalar, real_scalar)
  TYPE(Matrix), intent(out) :: mat_scalar
  Real(kind=dp), intent(in) :: real_scalar
  mat_scalar%element=real_scalar
end subroutine matrix_real_scalar

subroutine matrix_real_matrix(mat_matrix, real_matrix)
  TYPE(Matrix), intent(out) :: mat_matrix(:,:)
  Real(kind=dp), intent(in) :: real_matrix(:,:)
  mat_matrix(:,:,%)element=real_matrix(:,:,%)
end subroutine matrix_real_matrix

END MODULE MatrixMod
```

```
PROGRAM MatrixModEx
  use double
  use MatrixMod
  IMPLICIT NONE

  integer, parameter :: n=2
```

At first this doesn't seem that helpful but that is where interfaces come into play.
What this did was associate both of our new functions with the assignment operator. So now whenever we try to use = it will check to see if either of those functions work.