FORTRAN - DYNAMIC ARRAYS

A **dynamic array** is an array, the size of which is not known at compile time, but will be known at execution time.

Dynamic arrays are declared with the attribute **allocatable**.

For example,

```
real, dimension (:,:), allocatable :: darray
```

The rank of the array, i.e., the dimensions has to be mentioned however, to allocate memory to such an array, you use the **allocate** function.

```
allocate ( darray(s1,s2) )
```

After the array is used, in the program, the memory created should be freed using the **deallocate** function

```
deallocate (darray)
```

Example

The following example demonstrates the concepts discussed above.

```
program dynamic_array
implicit none

   !rank is 2, but size not known
   real, dimension (:,:), allocatable :: darray
   integer :: s1, s2
   integer :: i, j

   print*, "Enter the size of the array:"
   read*, s1, s2

   ! allocate memory
   allocate ( darray(s1,s2) )

   do i = 1, s1
      do j = 1, s2
         darray(i,j) = i*j
         print*, "darray(",i,",",j,") = ", darray(i,j)
      end do
   end do

   deallocate (darray)
end program dynamic_array
```

When the above code is compiled and executed, it produces the following result:

```
Enter the size of the array: 3,4
darray( 1 , 1 ) = 1.00000000
darray( 1 , 2 ) = 2.00000000
darray( 1 , 3 ) = 3.00000000
darray( 1 , 4 ) = 4.00000000
darray( 2 , 1 ) = 2.00000000
darray( 2 , 2 ) = 4.00000000
darray( 2 , 3 ) = 6.00000000
darray( 2 , 4 ) = 8.00000000
darray( 3 , 1 ) = 3.00000000
darray( 3 , 2 ) = 6.00000000
```

```
darray( 3 , 3 ) = 9.00000000
darray( 3 , 4 ) = 12.0000000
```

Use of Data Statement

The **data** statement can be used for initialising more than one array, or for array section initialisation.

The syntax of data statement is:

```
data variable / list / ...
```

Example

The following example demonstrates the concept:

```
program dataStatement
implicit none

   integer :: a(5), b(3,3), c(10),i, j
   data a /7,8,9,10,11/

   data b(1,:) /1,1,1/
   data b(2,:)/2,2,2/
   data b(3,:)/3,3,3/
   data (c(i),i=1,10,2) /4,5,6,7,8/
   data (c(i),i=2,10,2)/5*2/

   Print *, 'The A array:'
   do j = 1, 5
      print*, a(j)
   end do

   Print *, 'The B array:'
   do i = lbound(b,1), ubound(b,1)
      write(*,*) (b(i,j), j = lbound(b,2), ubound(b,2))
   end do

   Print *, 'The C array:'
   do j = 1, 10
      print*, c(j)
   end do

end program dataStatement
```

When the above code is compiled and executed, it produces the following result:

```
The A array:
7
8
9
10
11
The B array:
1  1  1
2  2  2
3  3  3
The C array:
4
2
5
2
6
2
7
2
8
```

```
2
```

Use of Where Statement

The **where** statement allows you to use some elements of an array in an expression, depending on the outcome of some logical condition. It allows the execution of the expression, on an element, if the given condition is true.

Example

The following example demonstrates the concept:

```
program whereStatement
implicit none

   integer :: a(3,5), i , j

   do i = 1,3
      do j = 1, 5
         a(i,j) = j-i
      end do
   end do

   Print *, 'The A array:'

   do i = lbound(a,1), ubound(a,1)
      write(*,*) (a(i,j), j = lbound(a,2), ubound(a,2))
   end do

   where( a<0 )
      a = 1
   elsewhere
      a = 5
   end where

   Print *, 'The A array:'
   do i = lbound(a,1), ubound(a,1)
      write(*,*) (a(i,j), j = lbound(a,2), ubound(a,2))
   end do

end program whereStatement
```

When the above code is compiled and executed, it produces the following result:

```
The A array:
0   1   2  3  4
-1  0   1  2  3
-2  -1  0  1  2
The A array:
5   5   5  5  5
1   5   5  5  5
1   1   5  5  5
```