CS 199 Computer Programming

Spring 2018

Lecture 8

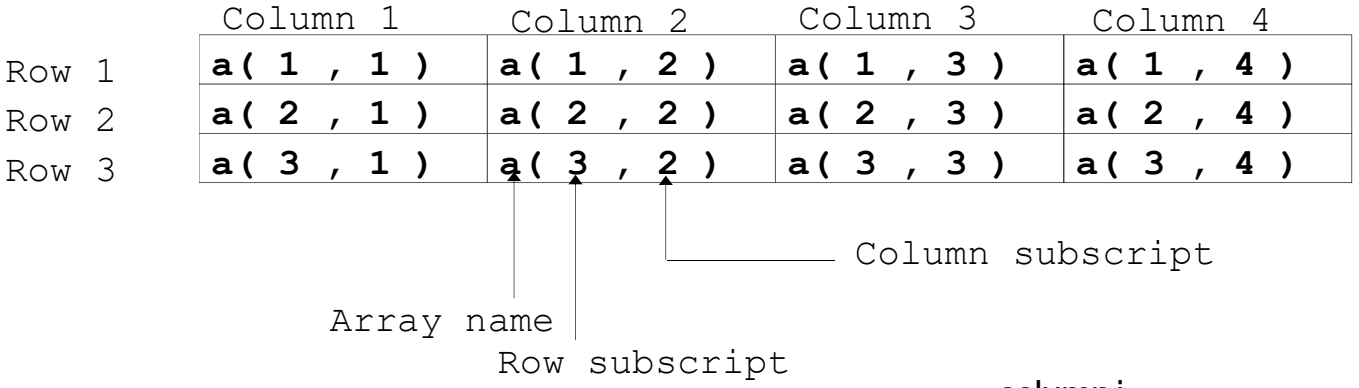
Two-dimension array

Multiple-Subscripted Arrays

- Arrays in FORTRAN can have multiple subscripts
- A common use is to represent
  - Tables with rows and columns
- **Double-Subscript arrays**
  - To identify a particular table element two subscripts must be specified
  - **A(i , j)**
    - **A** is the name of array
    - **i** and **j** are the subscripts that uniquely identify each element
  - Specify first index is row and second index is column
    - **(subscripts (i)=row & (j)=column)**
- Multiple subscript arrays can have more than two subscripts
  - FORTRAN compilers support at maximum 7 array subscripts

Multiple-Subscripted Arrays

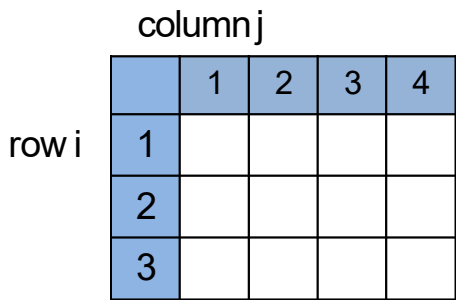
- **A double subscript array with 3 rows and 4 columns**



- **Declaring arrays**

Ex:

```
real :: a(3,4)
real :: c(0:3,-1:4)
```



Arrays

Visualization of Arrays

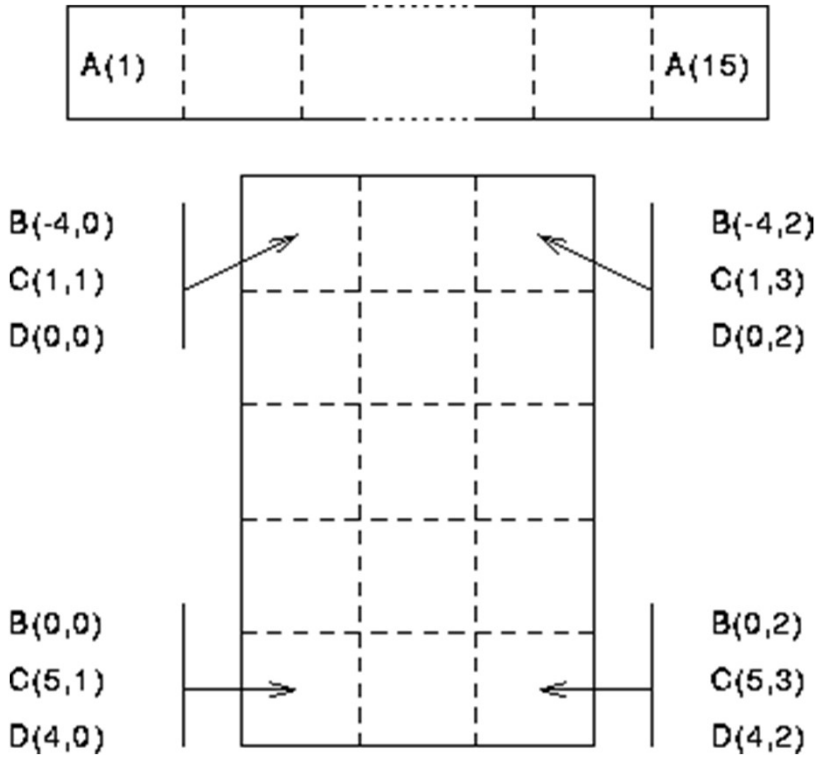
REAL :: A (15)

REAL :: B (-4:0,0:2)

REAL :: C (5,3)

REAL :: D(0:4,0:2)

•Individual array elements are denoted by *subscripting* the array name by an INTEGER, for example, A(7) element of A, or C(3,2), 3 elements down, 2 across.



Initialization with ASSIGNMENT statements

Initialize the whole array

```
integer :: istat(4,3)
do i =1,4
   do j =1,3
      istat(i,j) =5
   end do
end do
```

column j

row i

	1	2	3
1	5	5	5
2	5	5	5
3	5	5	5
4	5	5	5

Initialize the fourth row

```
integer :: istat(4,3)
   do j =1,3
      istat(4,j) =5
   end do
```

column j

row i

	1	2	3
1			
2			
3			
4	5	5	5

Initialize the first column

```
integer :: istat(4,3)
   do i =1,4
      istat(i,1) =5
   end do
```

column j

row i

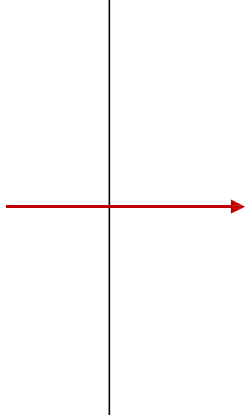
	1	2	3
1	5		
2	5		
3	5		
4	5		

Initialization with READ statements

```
integer :: istat1(4,3), istat2(4,3), istat3(4,3)
integer :: istat4(4,3), istat5(4,3)
do j=1,3
  do i=1,4
    read(*,*) istat1(i,j)
  end do
end do

read(*,*) ((istat2(i,j), i=1,4), j=1,3)

read(*,*) istat3
```



i

	1	2	3
1	1	2	3
2	1	2	3
3	1	2	3
4	1	2	3

Whole array operation

Most of the intrinsic functions operatecomponent-wise on arrays.

```
real, dimension(100,200) :: A,B,C
...
do j=1,200
do i=1,100
   C(i,j) =A(i,j) +B(i,j)
end do
end do
...
```

C= A+ B

```
real A(100), C(100)
...
do i=1,n
   C(i) =sin(A(i))
end do
...
```

C= sin(A)

Sum by Row

- To find the sum of row number 4 of matrix:

```
Do j = 1,3
   S= S + a(4,j)          ---- s= 18
```

- To find the sum of row number 2,3 of matrix:

```
Do i = 2,3
   Do j = 1,3
      S = s+ a(i,j)       ---- s=27
```

- To find the sum of row number 1,3 of matrix:

```
Do i = 1,3,2
   Do j = 1,3
      S = s+ a(i,j)       ---- s=24
```

```
Integer:: a(4,3),s=0,s
Do n=1,4
  Do m=1,3
      A(n,m)=n+m
  End do
End do
```

A=

2	3	4
3	4	5
4	5	6
5	6	7

Sum by Column

- To find the sum of column number 3 of matrix:

```
Do i = 1,4
    S= S + a(i,3)                    ---- s= 22
```

- To find the sum of column number 2,3 of matrix:

```
Do i = 1,4
    Do j = 2,3
        S = s+ a(i,j)                ---- s=40
```

- To find the sum of column number 1,3 of matrix:

```
Do i = 1,4
    Do j = 1,3,2
        S = s+ a(i,j)                ---- s=36
```

```
Integer:: a(4,3),s,s1(4)
Do n=1,4
  Do m=1,3
      A(n,m)=n+m
  End do
End do
```

A=

2	3	4
3	4	5
4	5	6
5	6	7

Example program

- **Following program is**
  - To keep track of students grades
  - A 4-by-3 array **studentGrades** is used (table)
  - Each Row of array represents a student
  - Each column represent a grade on one of three exams
- **Array manipulation in program is performed by four functions**
  - determines the lowest grade of any student for the semester
  - determines the highest grade of any student for the semester
  - determines a particular student's semester average

	Quiz1	Quiz2
Student0	95	85
Student1	89	80

FORTRANcode

```
integer :: studentgrades(4,3),min(4)
Read (*,*) studentgrades
Write (*,*) " the students grades="
 DO i = 1, 4 ! print row-by-row
WRITE(*,*) (studentgrades(i,j), j=1, 3)
min(i)=999999
END DO
Do i=1,4
 do j=1,3
    if (studentgrades(i,j) < min(i)) then
          min(i)= studentgrades(i,j)
      End if
End do
End do
```