Computers in Engineering COMP 208

Multi-Dimensional Arrays

Michael A. Hawker

A Two Dimensional Array

- A small hotel with four floors and six rooms on each floor could use a table with four columns and six rows to represent the number of occupants in each room:

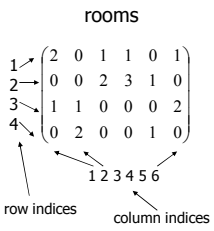
$$\begin{pmatrix} 2 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 2 & 3 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 2 \\ 0 & 2 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Multidimensional Arrays

- Think of a table with rows and columns.
- That forms a two dimensional array.
- A pile of these tables one on top of the other is a three dimensional array
- Fortran allows up to seven dimensions

Using Tables

- We must give the table a name to be able to reference it
- The rows and columns are referenced by a row index and a column index



Accessing Values

- To find the number of people occupying a room, we specify
  - the floor (the row index) and
  - the room (the column index)
- For example the number of occupants in the 5th room on the 2nd floor could be accessed by indexing as rooms(2,5)

Occupancy and Capacity

- A Hotel could be represented by two tables
  - One table might contain room occupancies
  - A second table might represent maximum room capacities

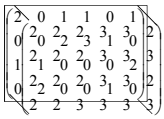
$$\begin{pmatrix} 2 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 2 & 3 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 2 \\ 0 & 2 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 2 & 2 & 2 & 3 & 3 & 2 \\ 2 & 2 & 2 & 3 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 & 2 \\ 2 & 2 & 3 & 3 & 3 & 3 \end{pmatrix}$$

occupancy capacity

Three Dimensional Arrays

We can combine the two tables into a single structure that represents the Hotel

This would form a three dimensional object



Accessing Values

- To access an individual cell, we need three indices
  - We select which table we want
    - the first – representing room occupancy or
    - the second – representing room capacity
  - We specify the floor (the row index) and
  - We specify the room (the column index)

Accessing Values

For example the number of occupants in the 5th room on the 2nd floor could be accessed by indexing as

```
Hotel(1,2,5)
```

The capacity of the same room would be

```
Hotel(2,2,5)
```

Declaring Multidimensional Arrays

- The declaration is similar to that of one dimensional arrays
- We specify a size for each dimension

```
INTEGER :: Hotel(2,4,6)
INTEGER :: Motel(2,2,18)
REAL :: Matrix(25,25)
```


Accessing Multidimensional Arrays

- We access the individual cells by specifying an index for each dimension.
- The index can be any integer expression.

```
Hotel(2,1,5)
Matrix(j,k)
Matrix(3,j+2*k)
```

- The values of the indices should be between one and the size of that dimension

Examples

- In the following examples, we see how to access two dimensional arrays

Processing Tables

- In going through all cells in a table, there are two methods
  - Cells can be accessed row by row
  - This is called row-major order
  - Cells can be accessed column by column
  - This is called column-major order
- Fortran stores tables internally in column major order

Array Input and Output

Read the array values row-by-row

```
INTEGER :: X(2,4)
. . .
DO I = 1, 2
  DO J = 1, 4
    READ(*,*) X(I,J)
  END DO
END DO
```


The Input Values

Assume the input is

```
4
8
3
9
7
5
2
6
```

Row Major Result

For the following input, one value per line

```
4  8  3  9  7  5  2  6
```

The resulting matrix, X, is

$$\begin{pmatrix} 4 & 8 & 3 & 9 \\ 7 & 5 & 2 & 6 \end{pmatrix}$$

Array Input and Output

Read the array values column-by-column

```
INTEGER :: X(2,4)
. . .
DO J = 1, 4
  DO I = 1, 2
    READ(*,*) X(I,J)
  END DO
END DO
```


Column Major Result

For the following input (the same as before) , one value per line

```
4  8  3  9  7  5  2  6
```

The resulting matrix, X, is

$$\begin{pmatrix} 4 & 3 & 7 & 2 \\ 8 & 9 & 5 & 6 \end{pmatrix}$$

Let's Compare

Input values:

```
4  8  3  9  7  5  2  6
```

$$\begin{pmatrix} 4 & 8 & 3 & 9 \\ 7 & 5 & 2 & 6 \end{pmatrix}$$

row major

$$\begin{pmatrix} 4 & 3 & 7 & 2 \\ 8 & 9 & 5 & 6 \end{pmatrix}$$

column major

Using Implied Loops

Read the array values row-by-row with an implied DO loop

```
INTEGER :: X(2,4)
. . .
DO I = 1, 2
  READ(*,*) (X(I,J), J = 1, 4)
END DO
```


Row Major (again)

Assume the input is

```
4 8 3 9 7 5 2 6
```

The resulting matrix, X, is

$$\begin{pmatrix} 4 & 8 & 3 & 9 \\ 7 & 5 & 2 & 6 \end{pmatrix}$$

Implied Do Loops (again)

Read the array values column-by-column with an implied DO loop

```
INTEGER :: X(2,4)
. . .
DO J = 1, 4
  READ(*,*) (X(I,J), I = 1, 2)
END DO
```


Column Major (again)

Assume the input is

```
4 8
3 9
7 5
2 6
```

The resulting matrix, X, is

$$\begin{pmatrix} 4 & 3 & 7 & 2 \\ 8 & 9 & 5 & 6 \end{pmatrix}$$

Nested Implied Loops

- We can use nested implied DO loops to read the values
- The following code reads values into an array in row major order
- The values can appear on one or more lines with no restrictions

```
INTEGER :: X(2,4)
. . .
READ(*,*) ((X(I,J), J = 1, 4), I = 1, 2)
```

Column Major Order

We can also use nested implied loops to read in column major order

```
INTEGER :: X(2,4)
. . .
READ(*,*) ((X(I,J), I = 1, 2), J = 1, 4)
```


Just plain read

- What about trying this?

```
INTEGER :: X(2,4)
. . .
READ(*,*) X
```

- This reads values one at a time into the cells of X, but where are they placed?
- Fortran stores arrays in ***Column Major Order***

More Matrix Processing

- So far we have just focused on reading values into a matrix
- Here are some more examples of applications where we process the cells of a two dimensional array

Initialize a Matrix to be the Identity Matrix

```
INTEGER :: SIZE = 20
INTEGER :: Ident (20, 20)
INTEGER :: i, j
DO i = 1, SIZE
  DO j = 1, SIZE
    IF (i == j) THEN
      Ident(i,i) = 1
    ELSE
      Ident(i,j) = 0
    END IF
  END DO
END DO
```


Sum Two Matrices

```
INTEGER :: SIZE = 20
REAL :: A(20,20), B(20,20), C(20,20)
INTEGER :: i, j

DO i = 1, SIZE
  DO j = 1, SIZE
    C(i,j) = A(i,j) + B(i,j)
  END DO
END DO
```


Transpose a Square Matrix

```
INTEGER :: SIZE = 20
REAL :: A(20,20), B(20,20), C(20,20)
INTEGER :: i, j
REAL :: Temp

DO i = 1, SIZE
  DO j = i+1, SIZE
    Temp = A(i,j)
    A(i,j) = A(j,i)
    A(j,i) = Temp
  END DO
END DO
```

What if j's initial value was 1 instead of i+1?

Multiply Square Matrices

```
INTEGER :: SIZE = 20
REAL :: A(20,20), B(20,20), C(20,20)
INTEGER :: i, j, k

DO i = 1, SIZE
  DO j = 1, SIZE
    C(i,j) = 0
    DO k = 1, SIZE
      C(i,j) = A(i,k)*B(k,j)
    END DO
  END DO
END DO
```


An Application

- A power generating station has four generators
- To determine productivity of each of the generators we sample the power supplied at six different time periods
- How do we represent the data?

Data Representation

- Use a two dimensional array with each column representing the power supplied by a generator
- Each row represents a time of measurement

```
INTEGER :: gens = 4
INTEGER :: samples = 6
REAL :: power(6, 4)
```

Power Output

We can calculate the power output by the entire plant at each sample time

```
REAL :: power_output(samples)
. . .
DO time = 1, samples
  power_output(time) = 0
  DO gen = 1, gens
    power_output(time) = &
          power_output(time)+ power(time,gen)
  END DO
END DO
```


Generator Output

We can calculate the average output of each generator

```
REAL :: gen_sum(gens), gen_avg(gens)
. . .
DO gen = 1, gens
  gen_sum(gen) = 0
  DO time = 1, samples
    gen_sum(gen) = &
          gen_sum(gen) + power(time,gen)
  END DO
  gen_avg(gen) = gen_sum(gen)/samples
END DO
```