Lecture 21

2D Array Use

In this lecture we will discuss following topics:

- Passing 2D to functions
- Passing 1D of 2D to functions
- Handling 2D column wise
- Matrix Operations (Add, Subtract, Multiply, Transpose)

Passing 2D to functions is similar to passing 1D. As argument we will simply use name of 2D array as we send a 1D array. As parameter we have to write data type followed by name of array followed by 2 square brackets. See example:

```
public static void fun(int a[][])
public static void fun(String s[][])
public static void fun(double d[][])
```

Passing 2D as 1D is required whenever we have to use a function having a 1D array as parameter. We can send a single row to such functions using array name and index of i^{th} row. We will discuss syntax but before that question is why we need to do this. Just think what type of data we can store in a 2D array? We may store score of different matches of 20 players or sale of 10 items for 12 months. In all such cases a single array can represent score of 1 player or score of all players in 1 match or sale of 1 item in 12 months or sales of all items in 1 month. Therefore in each case we may want to find average or max or min or sum of a player or of a match etc. Therefore all those functions may available for a 1D array can be used for each dimension of a 2D array. Now see an example to understand syntax as well:

```
class Test2D_1D{
  public static int findMax(int x[]){
    int i,max=x[0];
    for(i=1;i<size;i++)
      if (max<x[i])
        max=x[i];
    return max;
  }
  public static void main(String a[]){
    //scores of 2 players in matches
    int scores[]={{23,34,45,63,12},{33,64,55,23,22,20},{13,34,25,53,42}};
    System.out.println("Max of player 1:"+findMax(scores[0]);
    System.out.println("Max of player 2:"+findMax(scores[1]);
    System.out.println("Max of player 3:"+findMax(scores[2]);     }
}
```

Note to access each element of 2D array we need 2 indexes. As name of array is starting address of array, we learnt previously. Writing scores[0] means starting address of row first and scores[1] means starting address of row second. Expected output code is:

```
Max of player 1:63

Max of player 2:64

Max of player 3:53
```

Again I want to remind that we cannot access element of 2D array without 2 indexes. Access a 2D array without indexes or with single index is an error. However if we write array name without index inside function means we are sending starting address of array and if we write array name with single index mean we are sending starting address of i^{th} row of 2D array. In later case function can access only that particular row for which address is sent. Here we will do practice of 2D array and functions discussing some examples. Starting from a code to print row sum of a 2D array:

```
public static void printRowSum(int a[][]){
  int i,j,sum;
  for(i=0;i<a.length;i++){
    sum=0;
    for(j=0;j<a[i].length;j++)
      sum=sum+x[i][j];
    System.out.println ("Sum of row "+(i+1)+":"+sum);
}
```

The student once again note following important points. Nested loop is used to handle a 2D array. Variable sum is initialized by 0 inside inner loop because each row has separate sum. Print of sum is in outer loop not in inner loop because sum is to be printed after it is calculated in inner loop. Now assume with a minimal modification we want to print column sum instead of row sum. If you feel boring I want to state amusing note from a student that we will write “printColumnSum” in signature of function. Well that’s very right. The second change is we will write x[j][i] instead of x[i][j]. Also we will write a.length in place of a[i].length and in place of a.length we will not write a[i].length because we can’t do that. Rather we will simply write a[0].length which means code will work for only rectangular array having same number of columns in each row:

```
public static void printColumnSum(int a[][]){
  int i,j,sum;
  for(i=0;i<a[0].length;i++){
    sum=0;
    for(j=0;j<a.length;j++)
      sum=sum+x[j][i];
    System.out.println ("Sum of column "+(i+1)+":"+sum);
}
```

Lastly we will discuss some matrix operations because matrix requires 2D array because matrix has data in rows and columns of same type normally. A very simple function is to find transpose of a matrix. Function ***transpose*** will receive a matrix as input and return another matrix that will be have transpose of first matrix. In this operation we will declare a new 2D array than shift values from first 2D array to new 2D array from row to column wise:

```
public static int[][] transpose(int a[][]){
  int i,j;
  int b[][]=new int[a[0].length][a.length];
  for(i=0;i<a.length;i++)
    for(j=0;j<a[0].length;j++)
      b[j][i]=a[i][j];
  return b;
}
```

Here a[0].length is alright because we are dealing with rectangular array. Similar example is to add 2 matrices and return a third one having result:

```
public static int[][] add(int a[][],int b[][]){
  int i,j;
  int c[][]=new int[a.length][a[0].length];
  for(i=0;i<a.length;i++)
    for(j=0;j<a[0].length;j++)
      c[j][i]=a[i][j]+ b[i][j];
  return c;
}
```

Here we may use a.length or b.length because both have same size otherwise addition is not possible. Student must see how to multiply two matrices than try to implement multiply function.