# CSE 21
Intro to Computing II

Lecture 13 – Multi-dimensional Arrays

Announcement

- Lab#13 due before start of next lab
  - Type your answers in a text file and submit it as an attachment
- Final Exam
  - 12/10 (Saturday) at 11:30am
  - Cover everything
  - Open notes
  - Review next lecture
- Course evaluation online
  - Fill out by 12/9
  - Curve the class if more than 70% filled out
- Reading assignment
  - Chapter 5.9 of textbook

Review: 1D Arrays

- An array is a special object containing:
  - A group of contiguous memory locations that all have the same name and same type
  - A special (hidden) variable containing the number of elements in the array
- int[] a = new int[5];
- int[] b = a;

b

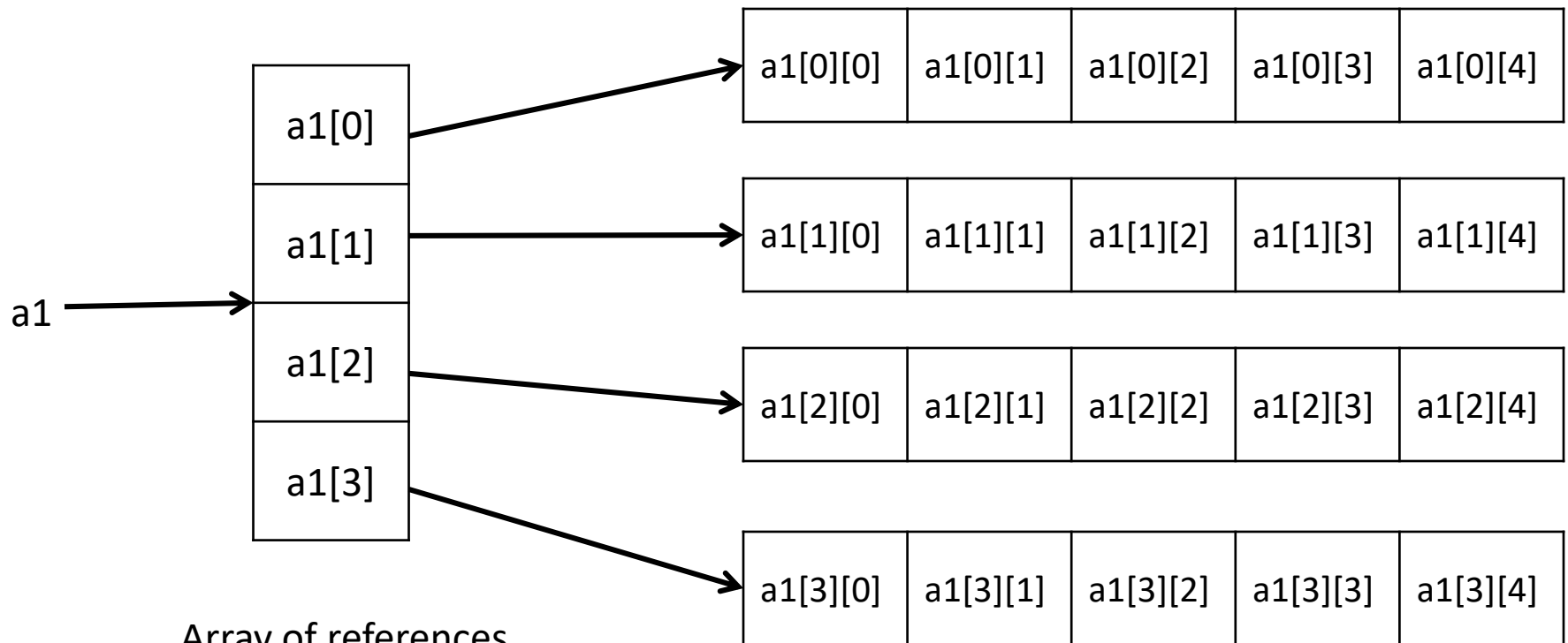| a[0] |
| a[1] |
| a[2] |
| a[3] |
| a[4] |

Array a

| 5 |

a.length
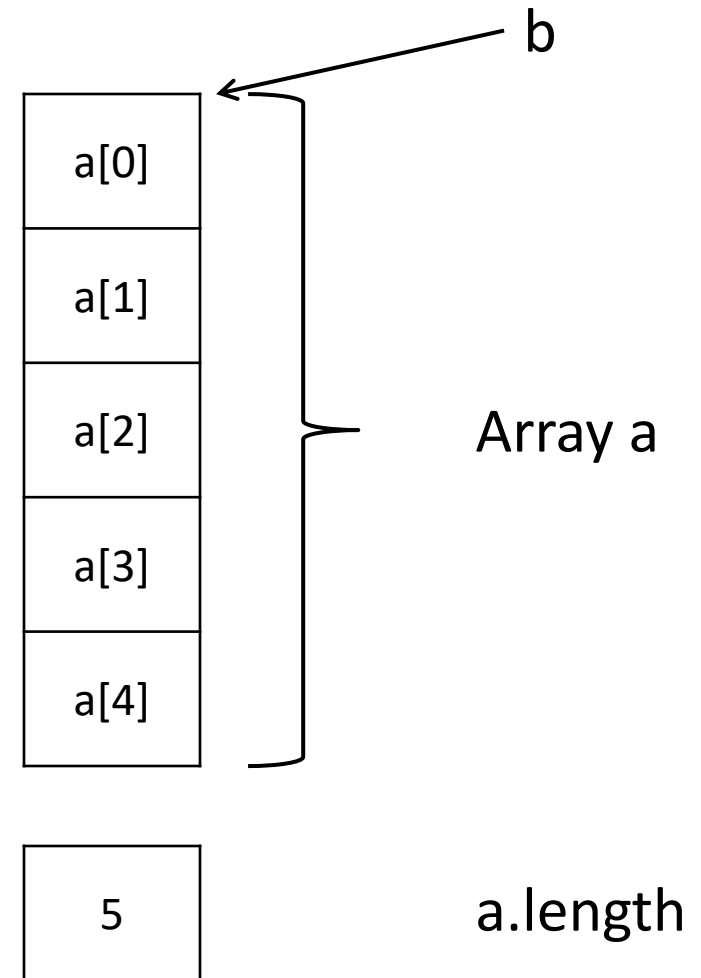
2D Arrays

- A two-dimensional (2D) array is an array of references (or pointers) to other arrays:
  - All arrays must be of the same type
- This creates a 2D data structure
- As matrices (in math), individual elements in the array are addressed with two subscripts, specifying the Row and Column (in that order) of the particular data item. (called row major)
- Memorize RC!!
  - (Radio Control, Royal Crown, Rice Cracker)

2D Arrays diagram

```
int[][] a1 = new int[4][5];
```

a1[0][0]	a1[0][1]	a1[0][2]	a1[0][3]	a1[0][4]

a1[0]

a1[1][0]	a1[1][1]	a1[1][2]	a1[1][3]	a1[1][4]

a1[1]

a1

a1[2]

a1[2][0]	a1[2][1]	a1[2][2]	a1[2][3]	a1[2][4]

a1[3]

a1[3][0]	a1[3][1]	a1[3][2]	a1[3][3]	a1[3][4]

Array of references
to other arrays

Arrays containing data

Declaring 2D Arrays

- We first declare an array of references to other arrays, then declare the arrays.
- Example:

```
double[][] a; //the actual array
a = new double[3][5];
```

- These steps can be combined on a single line, just like in 1D:

```
double[][] a = new double[3][5];
```

- 2D arrays may be initialized with nested array initializers

```
int[][] b = { {1,2,3}, {4,5,6} };
```

1st row   2nd row

Using 2D Arrays

- 2D arrays are used to represent data that is a function of two independent variables (or indices)
- A 2D array element is addressed using the array name followed by a integer subscript in brackets: **a[3][5]**
- Sizes of the arrays
  - a.length is the number of rows
  - a[0].length is the number of elements (cols) in row 0
  - a[1].length is the number of elements (cols) in row 1

Using 2D Arrays

▸ Example:

```
double[][] a = new double[3][5];
for ( r = 0; r < 3; r++ ) {
  for ( c = 0; c < 5; c++ ){
    a[r][c] = r*c; // Mult. table
  }
}
```

Indices	0	1	2	3	4
0					
1					
2					

Using 2D Arrays

- Example:

```
double[][] a = new double[3][5];
for ( r = 0; r < 3; r++ ) {
  for ( c = 0; c < 5; c++ ){
    a[r][c] = r*c; // Mult table
  }
}
```

Indices	0	1	2	3	4
0					
1					
2					

a[0][0]
a[0][1]
a[0][2]
a[0][3]
a[0][4]

a[1][0]
a[1][1]
a[1][2]
a[1][3]
a[1][4]

a[2][0]
a[2][1]
a[2][2]
a[2][3]
a[2][4]

Using 2D Arrays

▸ Example:

```
double[][] a = new double[3][5];
for ( r = 0; r < 3; r++ ) {
  for ( c = 0; c < 5; c++ ){
    a[r][c] = r*c; // Mult table
  }
}
```

Indices	0	1	2	3	4
0					
1					
2				?	

a[0][0]
a[0][1]
a[0][2]
a[0][3]
a[0][4]

a[1][0]
a[1][1]
a[1][2]
a[1][3]
a[1][4]

a[2][0]
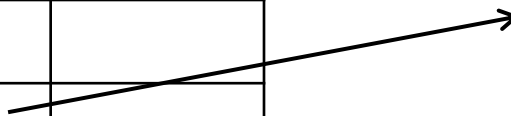a[2][1]
a[2][2]
a[2][3]
a[2][4]

Using 2D Arrays

▶ Example:

```
double[][] a = new double[3][5];
for ( r = 0; r < 3; r++ ) {
  for ( c = 0; c < 5; c++ ){
    a[r][c] = r*c; // Mult table
  }
}
```

Indices	0	1	2	3	4
0					
1					
2				2*3	

a[0][0]
a[0][1]
a[0][2]
a[0][3]
a[0][4]

a[1][0]
a[1][1]
a[1][2]
a[1][3]
a[1][4]

a[2][0]
a[2][1]
a[2][2]
a[2][3]
a[2][4]

Using 2D Arrays

▶ Example:

```
double[][] a = new double[3][5];
for ( r = 0; r < 3; r++ ) {
  for ( c = 0; c < 5; c++ ){
    a[r][c] = r*c; // Mult table
  }
}
```

Indices	0	1	2	3	4
0					
1			?		
2				6	

a[0][0]
a[0][1]
a[0][2]
a[0][3]
a[0][4]

a[1][0]
a[1][1]
a[1][2]
a[1][3]
a[1][4]

a[2][0]
a[2][1]
a[2][2]
a[2][3]
a[2][4]

Using 2D Arrays

▸ Example:

```
double[][] a = new double[3][5];
for ( r = 0; r < 3; r++ ) {
  for ( c = 0; c < 5; c++ ){
    a[r][c] = r*c; // Mult table
  }
}
```

Indices	0	1	2	3	4
0					
1			2		
2				6	

a[0][0]
a[0][1]
a[0][2]
a[0][3]
a[0][4]

a[1][0]
a[1][1]
a[1][2]
a[1][3]
a[1][4]

a[2][0]
a[2][1]
a[2][2]
a[2][3]
a[2][4]

Using 2D Arrays

▸ Example:

```
double[][] a = new double[3][5];
for ( r = 0; r < 3; r++ ) {
  for ( c = 0; c < 5; c++ ){
    a[r][c] = r*c; // Mult table
  }
}
```

Indices	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	6	8

a[0][0]
a[0][1]
a[0][2]
a[0][3]
a[0][4]

a[1][0]
a[1][1]
a[1][2]
a[1][3]
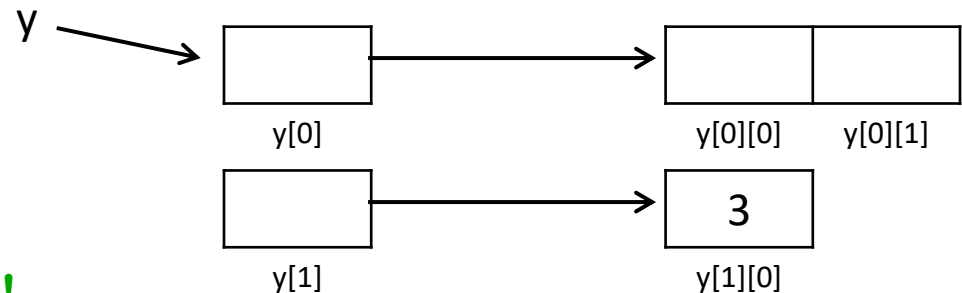a[1][4]

a[2][0]
a[2][1]
a[2][2]
a[2][3]
a[2][4]

2D Arrays: Rows with diff Columns

▸ Not all rows have to have the same # of cols:

```
int [][] x =
    new int [3][2];
//3 rows and 2 cols
```

x →

x[0]   x[0][0]   x[0][1]

x[1]   x[1][0]   x[1][1]

x[2]   x[2][0]   x[2][1]
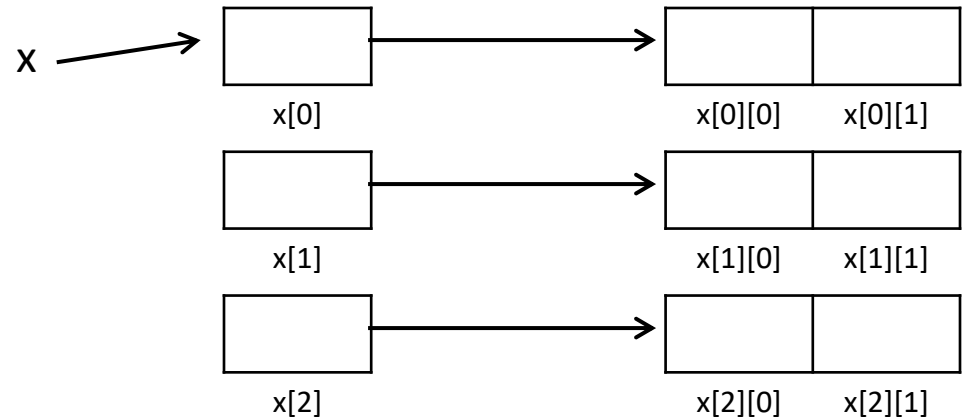
```
int [][] y =
    new int [2][];
y[0] = new int[2];
y[1] = new int[1];
y[1][0] = 3;
//2 rows: 2 and 1 cols!
```

y →

y[0]   y[0][0]   y[0][1]

y[1]   3
         y[1][0]

Higher-Order Arrays

▶ The same ideas that apply to 2D arrays can be extended for arrays of any order

▶ Example: the following statement creates a 3D array containing a total of 30 elements

```
double[][][] a = new double[3][5][2];
```

◦ The 1st element has the subscript range 0 to 2
◦ The 2nd element has the subscript range 0 to 4
◦ The 3rd element has the subscript range 0 to 1

▶ These are used to represent data that is a function of more than two independent variables (or indices)

◦ Example: a color image using 3 color spaces

File Input + 2D Array

```java
public static int[][] getInput(String filename) {
    int [][] arr = null;
    try {
        Scanner sc = new Scanner ( new FileReader(filename) );
        int row = sc.nextInt();
        int column = sc.nextInt();
        arr = new int[row][column];

        for (int i = 0 ; i < row; i++)
            for (int j = 0 ; j < column; j++)
                arr[i][j] = sc.nextInt();
        sc.close();
    } catch ( NoSuchElementException e){
        System.out.println(e);

    } catch (FileNotFoundException e) {
        System.out.println(e);
    }
    return arr;
}
```

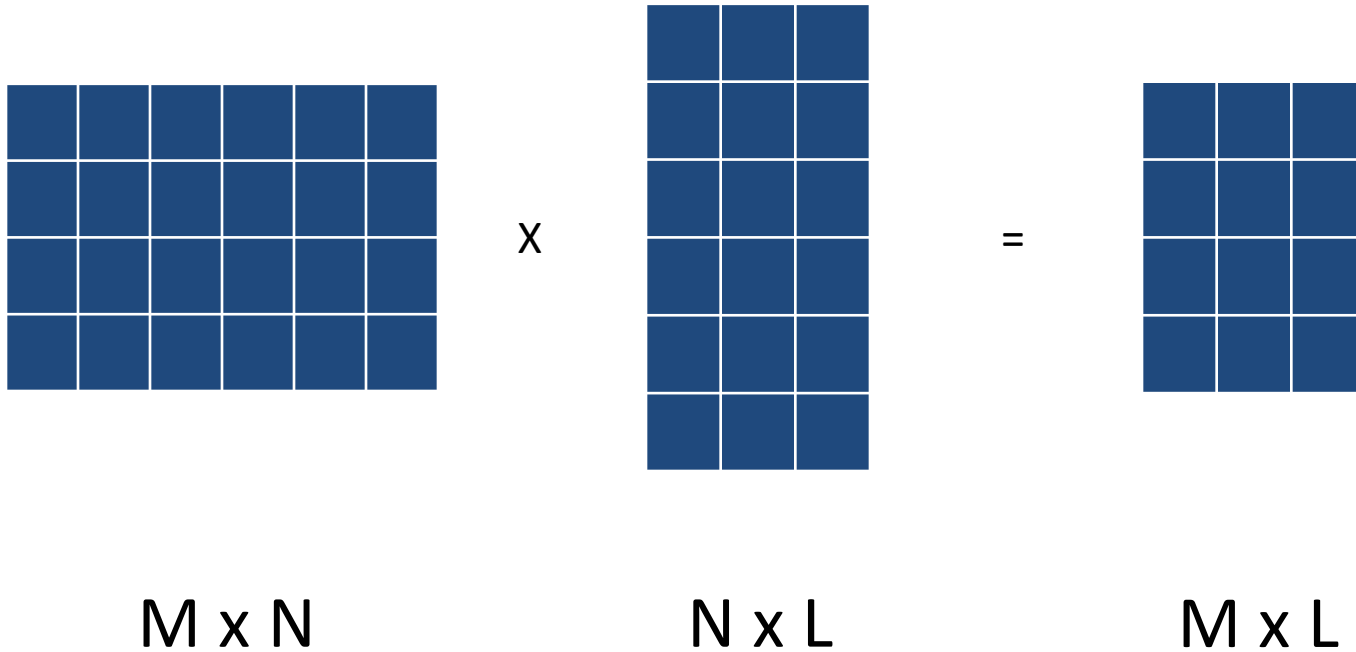# Square Matrix Multiplication

a	b
c	d

X

m	n
o	p

=

am+bo	an+bp
cm+do	cn+dp

General Matrix Multiplication

M x N   N x L   M x L

Matrix Multiply

```java
public static int[][] multiply(int[][] m1, int[][] m2) {
    int m1rows = m1.length;
    int m1cols = m1[0].length;
    int m2rows = m2.length;
    int m2cols = m2[0].length;
    if (m1cols != m2rows)
      throw new IllegalArgumentException("matrices don't match: " +
m1cols + " != " + m2rows);
    int[][] result = new int[m1rows][m2cols];

    // multiply
    for (int i=0; i<m1rows; i++)
      for (int j=0; j<m2cols; j++)
        for (int k=0; k<m1cols; k++)
          result[i][j] += m1[i][k] * m2[k][j];

    return result;
}
```

Example File Out

```java
String filename = "Result.txt";

try {
    FileWriter output = new FileWriter(filename);

    String ostr = "";
    for (int i = 0; i < arr2D.length; i++) {
        for (int j = 0; j < arr2D[0].length; j++) {
            System.out.print(ostr = (arr2D[i][j] + "\t"));
        output.write(ostr);
        }
        System.out.println();
        output.write("\r\n"); // Carriage return
    }
    output.close();
} catch (Exception e) {
    System.out.println(e);
}
```