Programming Refresher Workshop

Session 4

Mr Aaron Tan

Contents

- 2D Arrays
- 2D Array Initializer
- 2D Array Example
- Array of Arrays
- Ex 10: Maximum Sum of Path in Pyramid
- Matrices
- Matrix Addition

2D Arrays (1/2)

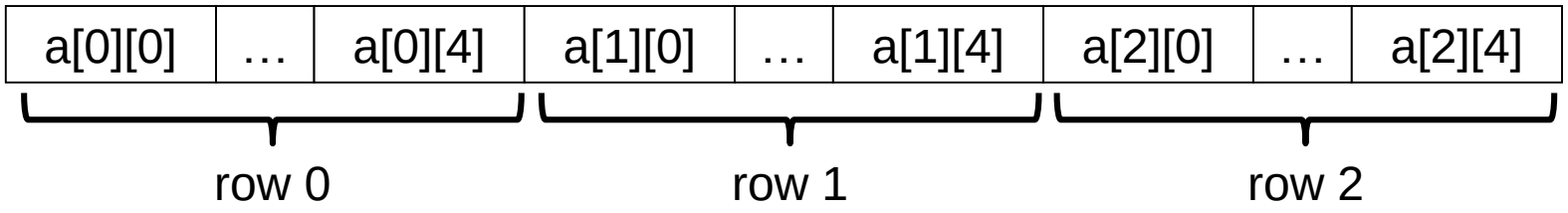
- In general, an array can have any number of dimensions
- Example of a 2-dimensional (2D) array:

In C

```
// array with 3 rows, 5 columns
int a[3][5];
a[0][0] = 2;
a[2][4] = 9;
a[1][0] = a[2][4] + 7;
```

	0	1	2	3	4
0	2				
1	16				
2					9

- Arrays are stored in row-major order



2D Arrays (1/2)

- Examples of applications:

$$\begin{Bmatrix} 3 & 8 & 2 \\ -5 & 2 & 0 \\ 1 & -4 & 9 \end{Bmatrix}$$

matrix[3][3]

	1	2	3	…	30	31
Jan	32.1	31.8	31.9		32.3	32.4
Feb	32.6	32.6	33.0	…	0	0
:						
Dec	31.8	32.3	30.9		31.6	32.2

Daily temperatures: temperatures[12][31]

Emily

	Ex1	Ex2	Ex3
Lab1	76	80	62
Lab2	60	72	48
Lab3	76	80	62
Lab4	60	72	48
Lab5	58	79	73

Zass

	Ex1	Ex2	Ex3
Lab1	59	68	60
Lab2	0	0	0
Lab3	67	71	75
Lab4	38	52	35
Lab5	78	86	82

Jerna

	Ex1	Ex2	Ex3
Lab1	79	75	66
Lab2	90	83	77
Lab3	81	73	79
Lab4	58	64	52
Lab5	93	80	85

Suise

	Ex1	Ex2	Ex3
Lab1	52	50	45
Lab2	57	60	63
Lab3	52	59	66
Lab4	33	42	37
Lab5	68	68	72

Students' lab marks: marks[4][5][3]

2D Array Initializer

```
// nesting one-dimensional initializers
int a[3][5] = { {4, 2, 1, 0, 0},
                {8, 3, 3, 1, 6},
                {0, 0 ,0, 0, 0} };

// the first dimension can be unspecified
int b[][5] = { {4, 2, 1, 0, 0},
               {8, 3, 3, 1, 6},
               {0, 0, 0, 0, 0} };

// initializer with implicit zero values
int d[3][5] = { {4, 2, 1},
                {8, 3, 3, 1, 6} };
```

What happens to the uninitialized elements?

2D Array Example

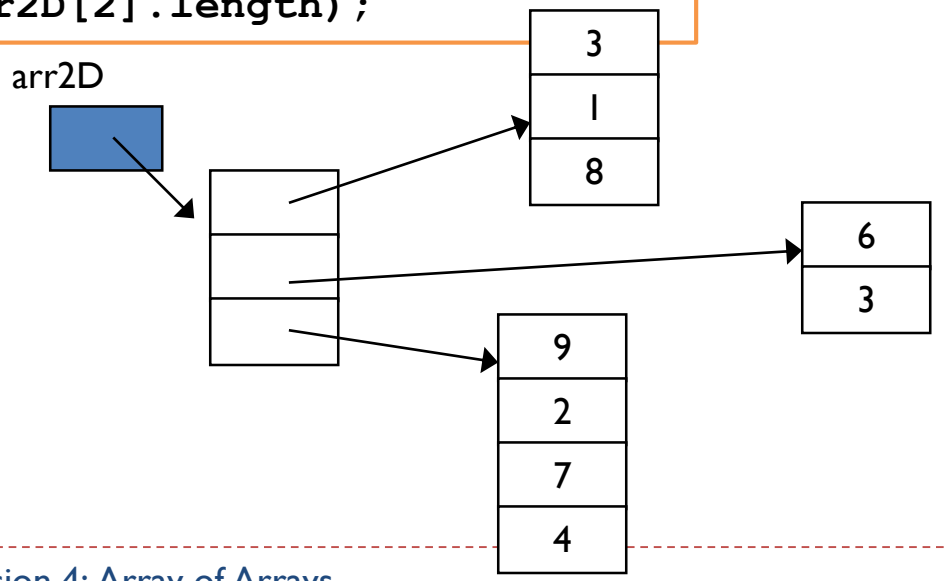
```
#include <stdio.h>
#define N 5    // number of columns in array
int sumArray(int [][N], int); // function prototype

int main(void) {
    int foo[][N] = { {3,7,1}, {2,1}, {4,6,2} };
    printf("sum is %d\n", sumArray(foo, 2));
    printf("sum is %d\n", sumArray(foo, 3));
    return 0;
}

// To sum all elements in arr
int sumArray(int arr[][N], int rows) {
    int i, j, total = 0;
    for (i = 0; i < rows; i++) {
        for (j = 0; j < N; j++) {
            total += arr[i][j];
        }
    }
    return total;
}
```

Second dimension must be specified; first dimension is not required.

Array of Arrays

- In some language, such as Java, a 2D array is actually an array of arrays.
- Hence, we may create a 2D array with rows different of different lengths.

```
int[][] arr2D = { {3,1,8}, {6,3}, {9,2,7,4} };
System.out.println(arr2D.length);
System.out.println(arr2D[0].length);
System.out.println(arr2D[1].length);
System.out.println(arr2D[2].length);
```



Ex 10: Maximum Sum of Path in Pyramid

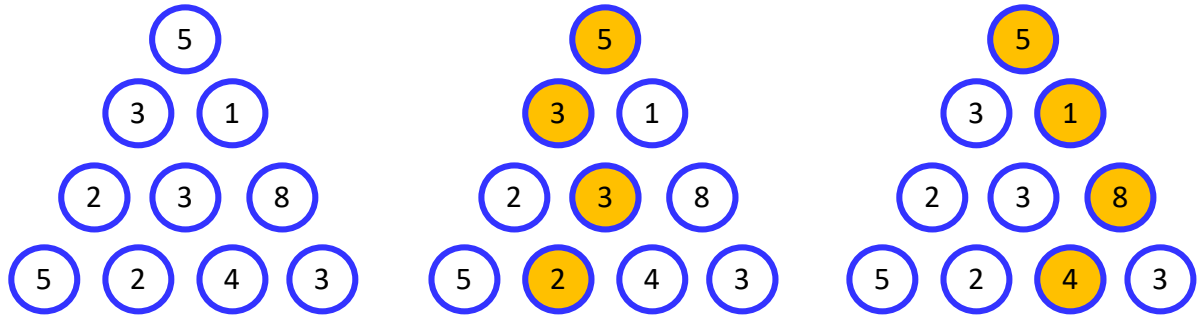


Figure 1. (a) A pyramid of integers. **(b)** A path with sum of 13. **(c)** A path with sum of 18.

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter number of rows: ");
int rows = sc.nextInt();

int[][] table = new int[rows][];
for (int i = 0; i < rows; i++)
  table[i] = new int[i+1];

System.out.println("Enter values for array: ");
for (int r = 0; r < table.length; r++)
  for (int c = 0; c < table[r].length; c++)
    table[r][c] = sc.nextInt();
```

Matrices

- A two-dimensional array where all rows have the same length is sometimes known as a matrix because it resembles that mathematical concept.
- A matrix *A* with *m* rows and *n* columns is represented mathematically in the following manner.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \cdots & & & \cdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

- Note that in implementing the matrix as an array in C/Java, the row number and column number start at 0 instead of 1.

Matrix Addition (1/4)

- To add two matrices, both must have the same number of rows and the same number of columns.
- To compute C = A + B, where A, B, C are matrices

$$c_{i,j} = a_{i,j} + b_{i,j}$$

- Example on 3×3 matrices:

$$\begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} + \begin{pmatrix} -1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 2 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 2 & 0 \\ 2 & 2 & 1 \\ 1 & 2 & 0 \end{pmatrix}$$

Matrix Addition (2/4)

```
#include <stdio.h>
#define MAX_ROW 10
#define MAX_COL 10

void scanMatrix(float [][MAX_COL], int *, int *);
void printMatrix(float [][MAX_COL], int, int);
void sumMatrix(float [][MAX_COL], float [][MAX_COL], float [][MAX_COL],
               int, int);

int main(void)
{
   float matrixA[MAX_ROW][MAX_COL]; // input matrix
   float matrixB[MAX_ROW][MAX_COL]; // input matrix
   float matrixC[MAX_ROW][MAX_COL]; // sum matrix
   int matrixArows, matrixAcols; // number of rows and columns for matrix A
   int matrixBrows, matrixBcols; // number of rows and columns for matrix B

   printf("Matrix A:\n");
   scanMatrix(matrixA, &matrixArows, &matrixAcols);
   printf("Matrix B:\n");
   scanMatrix(matrixB, &matrixBrows, &matrixBcols);
```

Matrix Addition (3/4)

```
   if ((matrixArows == matrixBrows) && (matrixAcols == matrixBcols)) {
      sumMatrix(matrixA, matrixB, matrixC, matrixArows, matrixAcols);
      printf("Sum matrix:\n");
      printMatrix(matrixC, matrixArows, matrixAcols);
   }
   else
      printf("Unmatched dimensions; cannot be added.\n");

   return 0;
}
```

```
// To read values into mtx
void scanMatrix(float mtx[][MAX_COL], int *row_size_p, int *col_size_p) {
   int row, col;

   printf("Enter number of rows and columns: ");
   scanf("%d %d", row_size_p, col_size_p);
   printf("Enter values for matrix:\n");
   for (row = 0; row < *row_size_p; row++)
      for (col = 0; col < *col_size_p; col++)
         scanf("%f", &mtx[row][col]);
}
```

Matrix Addition (4/4)

```
// To print values of mtx
void printMatrix(float mtx[][MAX_COL], int row_size, int col_size) {
   int row, col;

   for (row = 0; row < row_size; row++) {
      for (col = 0; col < col_size; col++)
         printf("%.2f\t", mtx[row][col]);
      printf("\n");
   }
}
```

```
// To sum mtxA and mtxB to obtain mtxC
void sumMatrix(float mtxA[][MAX_COL], float mtxB[][MAX_COL],
               float mtxC[][MAX_COL], int row_size, int col_size) {
   int row, col;









}
```

The End