Chapter 03 Multi-dimensional Arrays

The preceding chapter introduced how to use one-dimensional arrays (1D arrays) to store linear collections of elements. You can use a two-dimensional array (2D array) to store a matrix or a table. A 2D array is actually an array in which each element is a 1D array. Data in a table or a matrix can be stored in a two-dimensional array (2D array).

For example, the following table, which provides the distances between cities, can be stored in a array named distances.

Distance Table (in miles)

	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	1,375	967	1,087
Boston	983	0	214	1,102	1,505	1,723	1,842
New York	787	214	0	888	1,549	1,548	1,627
Atlanta	714	1,102	888	0	661	781	810
Miami	1,375	1,505	1,549	661	0	1,426	1,187

The data in the table above can be stored in an array named `distances` as below:

```
double[][] distances = {
    {0, 983, 787, 714, 1375, 967, 1087},
    {983, 0, 214, 1102, 1763, 1723, 1842},
    {787, 214, 0, 888, 1549, 1548, 1627},
    {714, 1102, 888, 0, 661, 781, 810},
    {1375, 1763, 1549, 661, 0, 1426, 1187}
};
```

1. 2D Array Basics

2.1 Declaring Variables of 2D Arrays

The syntax for declaring a two-dimensional array is as follows:

```
type[][] arrayVar;
```

or

```
type arrayVar[][]; // not preferred
```

As an example, here is how you would declare a two-dimensional array variable matrix of int values:

```
int[][] matrix;
```

or

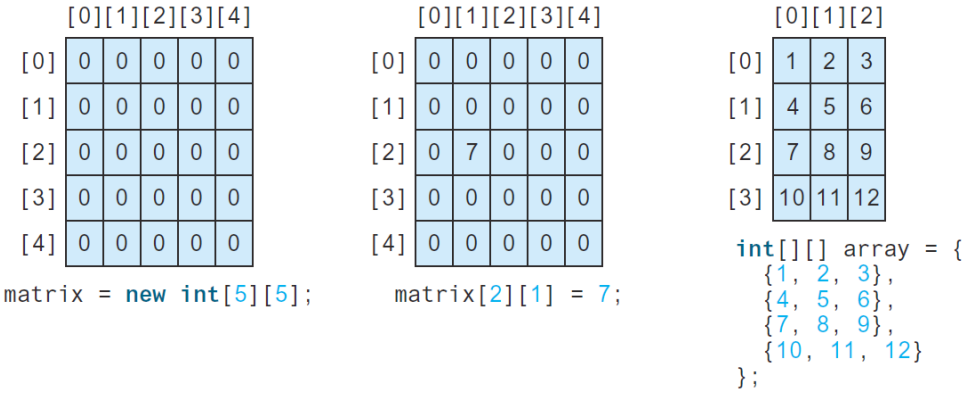
```
int matrix[][]; // not preferred
```

2.2 Creating Two-Dimensional Arrays

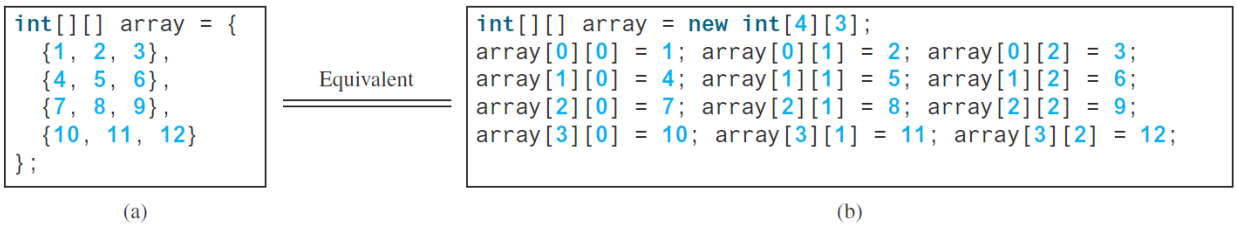
You can create a two-dimensional array of 5-by-5 int values and assign it to matrix using this syntax:

```
matrix = new int[5][5];
```

Two subscripts are used in a two-dimensional array: one for the row, and the other for the column.



The following code in (a) creates an array with the specified initial values. This is equivalent to the code in (b).



(a) (b)

2.3 Obtaining the Lengths of 2D Arrays

Since a 2D array is actually an array in which each element is a 1D array, we can use `length` to get the length of the array. For example, suppose that `x = new int[3][4];`

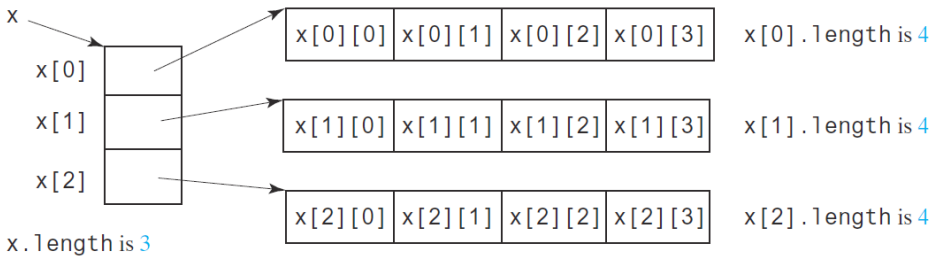


FIGURE 8.2 A two-dimensional array is a one-dimensional array in which each element is another one-dimensional array.

2.4 Ragged Arrays

Each row in a two-dimensional array is itself an array. Thus, the rows can have different lengths. An array of this kind is known as a ragged array. Here is an example of creating a ragged array:

Example 01: Using ragged an array.

```
class Example {
    public static void main(String[] args) {
        int[][] array = {
            {10, 983},
            {98, 0, 214, 1102},
            {78, 214, 0, 88, 15},
            {71, 11},
            {75, 63, 19},
            {96, 17},
        };
        System.out.print(array[2][4]);
        //System.out.print(array[1][4]); // Error: Index 4 out of bounds for
                                         // length 4
    }
}
```

Output:

```
15
```

In high level languages such as Java, there are functionalities which prevent you from accessing array out of bound by generating an exception such as java.lang.ArrayIndexOutOfBoundsException. But in case of C/C++, there is no such functionality, so programmer need to take care of this situation.

3. Processing 2D Arrays

Nested **for** loops are often used to process a two-dimensional array.

Example 02: Initialize the array with user input values.

```
import java.util.Scanner;

class Example {
    public static void main(String[] args) {

        int[][] matrix = new int[3][4];
        Scanner input = new Scanner(System.in);

        System.out.println("Enter " + matrix.length + " rows and " +
                           matrix[0].length + " columns: ");
        // Input the array
        for (int row = 0; row < matrix.length; row++) {
            for (int col = 0; col < matrix[row].length; col++) {
                matrix[row][col] = input.nextInt();
            }
        }
        input.close();

        System.out.println();

        //Display the array
        for (int row = 0; row < matrix.length; row++) {
            for (int column = 0; column < matrix[row].length; column++) {
                System.out.printf("%-3d ", matrix[row][column]);
            }
            System.out.println();
        }
    }
}
```

Output:

```
1  2  3  4
5  6  7  8
9  10 11 12
```

Example 03: Initialize the array with random values between 0 and 99.

```
class Example {
    public static void main(String[] args) {

        int[][] matrix = new int[3][4];

        //Initialize arrays with random values between 0 and 99
        for (int row = 0; row < matrix.length; row++) {
            for (int column = 0; column < matrix[row].length; column++) {
                matrix[row][column] = (int)(Math.random() * 100);
            }
        }

        //Display the array
        for (int row = 0; row < matrix.length; row++) {
            for (int column = 0; column < matrix[row].length; column++) {
                System.out.printf("%-3d ", matrix[row][column]);
            }
            System.out.println();
        }
    }
}
```

Output:

```
95  39  23  81
81  47  91  99
2   19  72  23
```

4. Shuffling the Elements in 2D Array

To shuffle the elements in 2D array, `matrix`, for each element `matrix[i][j]`, randomly generate indices `i1` and `j1` and swap `matrix[i][j]` with `matrix[i1][j1]`, as follows:

Example 04: Shuffle the elements in 2D array

```
import java.util.Arrays;

class Example {
    public static void main(String[] args) {
        int[][] array2D = {{1,  2,  3,  4},{5,  6,  7,  8},{9, 10, 11, 12}};

        // Shuffle the array
        for (int i = 0; i < array2D.length; i++) {
            for (int j = 0; j < array2D[i].length; j++) {
                int i1 = (int)(Math.random() * array2D.length);
                int j1 = (int)(Math.random() * array2D[i].length);

                // Swap array2D[i][j] with array2D[i1][j1]
                int temp = array2D[i][j];
                array2D[i][j] = array2D[i1][j1];
                array2D[i1][j1] = temp;
            }
        }

        //Display the array
        System.out.print(Arrays.deepToString(array2D));
    }
}
```

Output:

```
[[12, 3, 6, 1], [11, 7, 2, 9], [10, 5, 4, 8]]
```

The `Arrays.deepToString()` is a method defined in Arrays class that converts multidimensional arrays to strings

4. Multi-dimensional Arrays

A two-dimensional array is an array of one-dimensional arrays, and a three-dimensional array is an array of two-dimensional arrays, and so on. You can create n-dimensional arrays for any integer n.

Example 13: For example, you can use a three-dimensional array to store exam scores for a class of six students with five exams, and each exam has two parts (multiple-choice and essay).

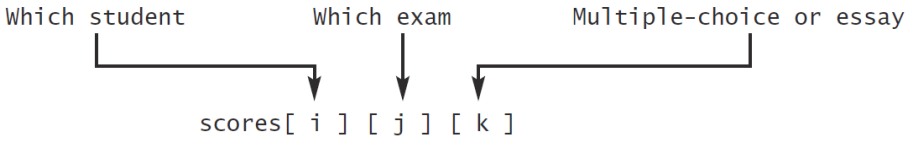
```
import java.util.Arrays;

class Example {
    public static void main(String[] args) {
        double[][][] scores = {
            {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
            {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
            {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
            {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
            {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
            {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}}
        };
        System.out.println(Arrays.deepToString(scores[0]));
        System.out.println(Arrays.toString(scores[0][1]));
        System.out.println(scores[0][1][0]);
        System.out.println(scores[0][1][1]);
        System.out.println(scores.length);
        System.out.println(scores[0].length);
        System.out.println(scores[0][0].length);
    }
}
```

Output:

```
[[7.5, 20.5], [9.0, 22.5], [15.0, 33.5], [13.0, 21.5], [15.0, 2.5]]
[9.0, 22.5]
9.0
22.5
6
5
2
```

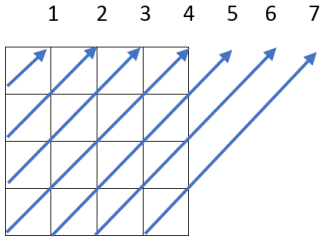
`scores[0][1][0]` refers to the multiple-choice score for the first student's second exam, which is `9.0`. `[0][1][1]` refers to the essay score for the first student's second exam, which is `22.5`. This is depicted in the following figure:



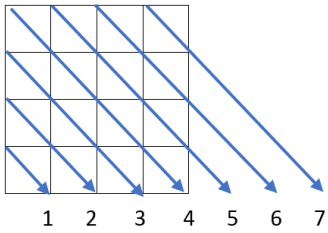
Exercises

Write the following programs:

1. Iterate over a 2D array diagonally right up using pointers.



2. Iterate over a 2D array diagonally right down using pointers.



3. Generates a random 10-by-10 2D array of uppercase letters (A to Z).
4. *(Sort a vector of points on y-coordinates)* Write a program to sort a vector of points on their y-coordinates. Each point is a vector of two values for x- and y-coordinates.

   For example, the points {{4, 2}, {1, 7}, {4, 5}, {1, 2}, {3, 1}, {4, 1}} will be sorted into {{3, 1}, {4, 1}, {1, 2}, {4, 2}, {4, 5}, {1, 7}}.

5. (Algebra: add two matrices) Write a program to add two matrices. In order to be added, the two matrices must have the same dimensions and the same or compatible types of elements.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & a_{13} + b_{13} \\ a_{21} + b_{21} & a_{22} + b_{22} & a_{23} + b_{23} \\ a_{31} + b_{31} & a_{32} + b_{32} & a_{33} + b_{33} \end{pmatrix}$$

   Write a test program that prompts the user to enter two 3 x 3 matrices and displays their sum. Here is a sample run:

```
Enter matrix1: 1 2 3 4 5 6 7 8 9 ↵Enter
Enter matrix2: 0 2 4 1 4.5 2.2 1.1 4.3 5.2 ↵Enter
The matrices are added as follows:
 1.0 2.0 3.0       0.0 2.0 4.0     1.0 4.0 11.0
 4.0 5.0 6.0   +   1.0 4.5 2.2  =  5.0 11.5 8.2
 11.0 8.0 11.0     1.1 4.3 5.2     8.1 12.3 14.2
```

6. *(Points nearest to each other)* Write a program that finds two points in a three-dimensional space nearest to each other. Use a two-dimensional array to represent the points. Test the program using the following points:

```
double[][] points = { {-1, 0, 3}, {-1, -1, -1}, {4, 1, 1}, {2, 0.5, 9},
   {3.5, 2, -1}, {3, 1.5, 3},{-1.5, 4, 2}, {5.5, 4, -0.5}
};
```

   The formula for computing the distance between two points is: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$

7. *(Guess the capitals)* Write a program that repeatedly asks the user to enter a capital city for a country. Upon receiving the user input, the program reports whether the answer is correct. Assume that 10 countries and their capital cities are stored in a 2D array. The program asks the user to answer all the countries' capital cities and displays the total correct count. The user's answer is not case-sensitive. And the order of the questions is randomly. Implement the program using a 2D array to represent the data in the following table:

Cambodia	Phnom Penh
Thailand	Bangkok
China	Beijing
Japan	Tokyo
India	Delhi
Malaysia	Kuala Lumpur
…	…

Here is a sample run:

```
What is the capital of Alabama? Montogomery ↵Enter
The correct answer should be Montgomery
What is the capital of Alaska? Juneau ↵Enter
Your answer is correct
What is the capital of Arizona? ...
...
The correct count is 35
```

8. *(Compute the weekly hours for each employee)* A table below shows the seven-day work hours of eight employees. Write a program that displays employees in decreasing order of the total hours.

	Su	M	T	W	Th	F	Sa
Employee 0	2	4	3	4	5	8	8
Employee 1	7	3	4	3	3	4	4
Employee 2	3	3	4	3	3	2	2
Employee 3	9	3	4	7	3	4	1
Employee 4	3	5	4	3	6	3	8
Employee 5	3	4	4	6	3	4	4
Employee 6	3	7	4	8	3	8	4
Employee 7	6	3	5	9	2	7	9

9. *(Financial tsunami)* Banks loan money to each other. If a bank goes bankrupt, it may not be able to pay back the loan. A bank's total asset is its current balance plus its loans to other banks. Figure 8.7 is a diagram that shows five banks. The current balance of the bank 0 is 25, and its total asset is 25 + 100.5 + 320.5 = 446 million dollars. The directed edge from node 0 to node 1 indicates that bank 0 loans 100.5 million dollars to bank 1.

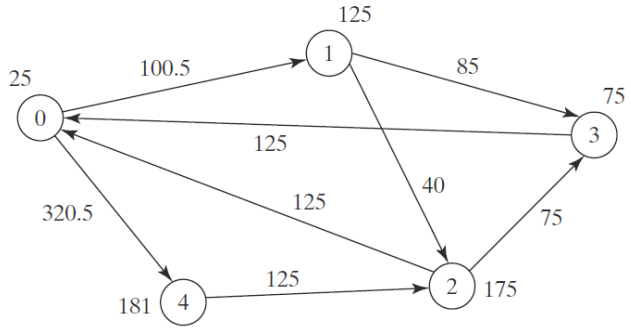


FIGURE 8.7 Banks loan money to each other.

If a bank's total asset is under a certain limit, the bank is unsafe. If a bank is unsafe, the money it borrowed cannot be returned to the lender, and the lender cannot count the loan in its total asset. Consequently, the lender may also be unsafe, if its total asset is under the limit.

Write a program that asks a user to enter a limit which is the minimum asset for keeping a bank safe, then displays all unsafe banks. Let's try giving the limit of 201 million and 300 million, and see the results.

10. Ask a user for scores of 8 students divided into 2 groups of 4 students in each. Each student studies 3 subjects, so each of them gets 3 scores. You must store those scores in a three-dimensional array first, then access the elements of the array to calculate the total score of each student, the total score of each group of students, and the total score of all students. Then, display the result in a tabular format, for example:

Group 1:

Name	Maths	Physics	Chemistry	Total
Student 1	50	30	70	150
Student 2	60	40	80	180
Student 3	70	50	90	210
Student 4	80	60	100	240
			Total Score:	780

Group 2:

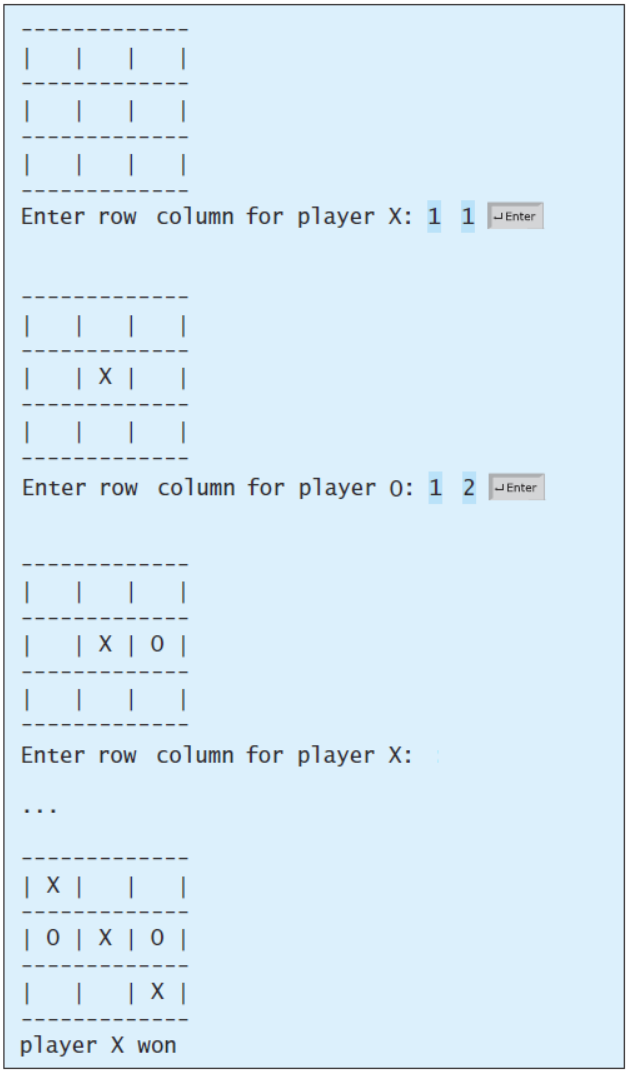
Name	Maths	Physics	Chemistry	Total
Student 5	55	35	75	165
Student 6	65	45	85	195
Student 7	75	55	95	225
Student 8	85	65	90	240
			Total Score:	825

11. *(Maths Quiz)* Make another quiz program that asks five questions by randomly generating two integer numbers between 1 to 20 to make the question (What is [num1] + [num2]?). There must be four operation: +, -, * and /. And if the result is floating-point number, make it only two decimal places. Ask the user to enter the answer. If they get it right add a point to their score. At the end of the quiz, tell them how many they got correct out of five and the details. Here is a sample run:

```
Question 1: What is 5 + 10? 15      Enter
Question 2: What is 2 - 10? 5       Enter
Question 3: What is 4 / 9? 0.44     Enter
Question 4: What is 17 * 9? 10      Enter
Question 5: What is 20 + 15? 8      Enter

You get 2 correct answers out of 5. The detail is below:
Question 1:  5 + 10 = 15     Correct.
Question 2:  2 - 10 = 5      Incorrect. The correct answer is -8
Question 3:  4 / 9 = 0.44    Correct.
Question 4:  17 * 20 = 10    Incorrect. The correct answer is 340
Question 5:  20 / 15 = 8     Incorrect. The correct answer is 1.33
```

12. *(TicTacToe)* The program asks the first player to enter an X token, and then asks the second player to enter an O token. Whenever a token is entered, the program redisplays the board on the console and determines the status of the game (win, draw, or unfinished). Here is a sample run:



Reference

[1] Y. Daniel Liang. 'Introduction to Java Programming', 11e – 2019

[2] https://www.programiz.com/java-programming