7.1Two-Dimensional Array Basics

7.1.1 Declaring and Creating Two-Dimensional Arrays

The syntax for declaring a two-dimensional array:

elementType [ ][ ] arrayRefVar;
or
elementType arrayRefVar [ ][ ];

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

int [ ][ ] matrix;
or
int matrix [ ][ ];

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

matrix = **new int**[row][column];

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

The matrix array is a *two-dimensional array* because two indices (one for the row and another for the column) are used to refer to an array element.

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

As in a one-dimensional array, the index for each subscript is of the **int** type and starts from **0**, as shown in Figure 8.1(a).

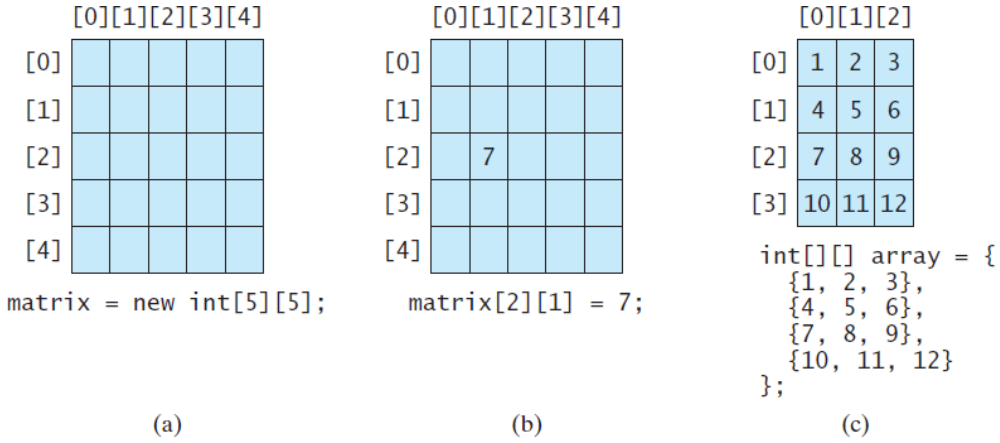


FIGURE 8.1 The index of each subscript of a two-dimensional array is an ***int*** value, starting from **0**.

To assign the value **36.50** to a specific element at row **2** and column **1**, as shown in Figure 8.2, you can use the following:

double [ ] [ ] matrix = new double [4] [5]

matrix [**2**] [**1**] = **36.50** ;

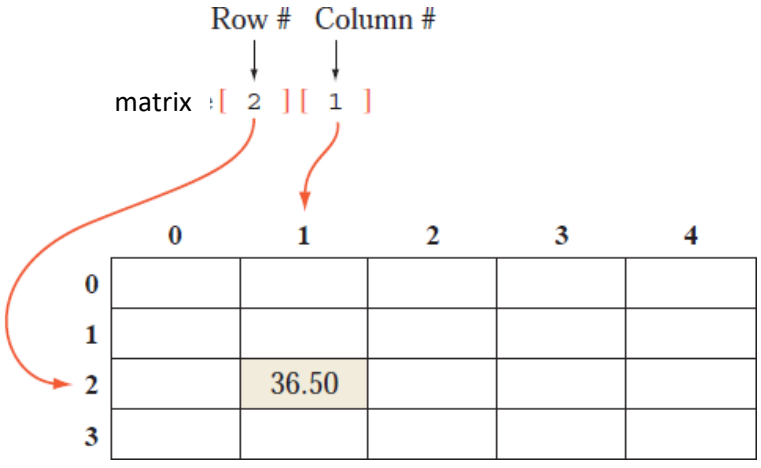


Figure 8.2 illustrates how the two indices are used to access an array element of a two-dimensional array.

You can also use an array initializer to declare, create, and initialize a two-dimensional array. For example, the following code in (a) creates an array with the specified initial values, as shown in Figure 8.3 (a). This is equivalent to the code in (b).

```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

(a)

Equivalent

```
int[][] array = new int[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

(b)

7.1.2 Obtaining the Lengths of Two-Dimensional Arrays

A two-dimensional array is actually an array in which each element is a one-dimensional array. The length of an array **x** is the number of elements in the array, which can be obtained using ***x.length*** .

x[0], **x[1]**, and **x[x.length-1]** are arrays. Their lengths can be obtained using **x[0].length**, **x[1].length**, and **x[x.length-1].length**.

For example, suppose

x = new int[3][4],

x.length is **3**,

and

x[0].length is 4

x[1].length is 4

x[2].length is 4

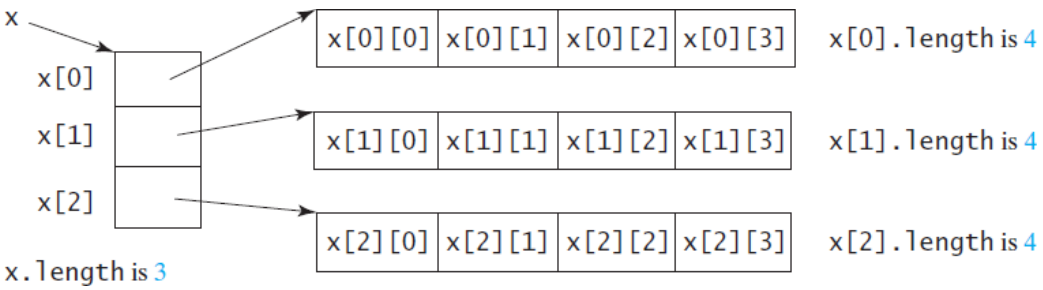
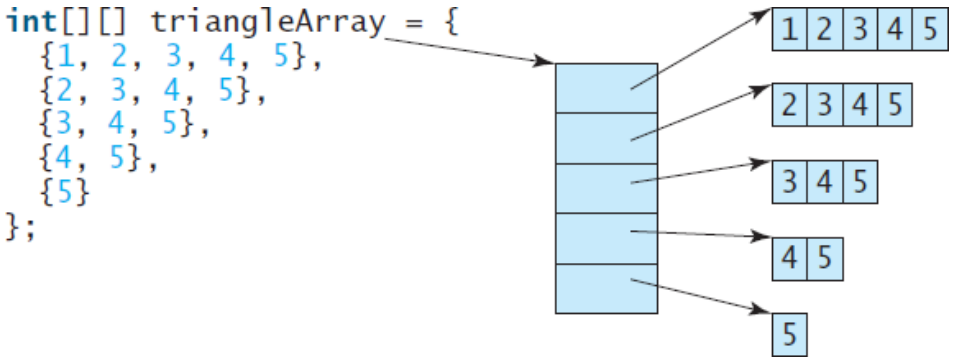


FIGURE 8.3 A two-dimensional array is a one-dimensional array in which each element is another one-dimensional array.

7.1.3 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:



As can be seen, **triangleArray[0].length** is 5, **triangleArray[1].length** is 4, **triangleArray[2].length** is 3, **triangleArray[3].length** is 2, and **triangleArray[4].length** is 1.

you can create a ragged array using the syntax that follows:

```
int [ ] [ ] triangleArray = new int[5][ ] ;
triangleArray[0] = new int[5];
triangleArray[1] = new int[4];
triangleArray[2] = new int[3];
triangleArray[3] = new int[2];
triangleArray[4] = new int[1];
```

7.2 Processing Two-Dimensional Arrays

Suppose an array **matrix** is created as follows:

```
int [ ][ ] matrix = new int [10][10];
```

Here are some examples of processing two-dimensional arrays:

1. (*Initializing arrays with input values*) The following loop initializes the array with user input values:

```
void read ( int [ ][ ] matrix) {
Scanner input = new Scanner (System.in);
System.out.println("Enter elements of array: ");

  for (int i = 0; i < matrix.length; i++) {
       for (int j = 0; j < matrix [i].length ; j++) {
           matrix[i][j] = input.nextInt( );
        }
  }
```

2. (*Initializing arrays with random values*) The following loop initializes the array with random values between 0 and 99:

```
void randomValue ( int [ ][ ] matrix){

  for (int row = 0; row < ; row++) {
       for (int column = 0; column < matrix[row].length ; column++) {

           matrix[row][column] = (int)(Math.random( ) * 100);
       }
  }
```

3. (*Printing arrays*) To print a two-dimensional array, you have to print each element in the array using a loop like the following:

```
void print ( int [ ][ ] matrix){

  for (int row = 0; row < matrix.length; row++) {
       for (int column = 0; column < matrix[row].length ; column++) {
           System.out.print ( matrix [row][column] + " ");
       }
   System.out.println ( );
  }
```

4. (*Summing all elements*) Use a variable named **total** to store the sum. Initially **total** is **0**. Add each element in the array to **total** using a loop like this:

```
int sum ( int [ ][ ] matrix){
  int total = 0;
  for (int i = 0; row < i.length; i++) {
       for (int j = 0; j < matrix[i].length; j++) {
            total +=  matrix [i][j];
```

```
        }
      }
     return total;
     }
```

5. (*Summing elements by column*) For each column, use a variable named **total** to store its sum. Add each element in the column to **total** using a loop like this:

```
void SumOfEachcolumn ( int [ ][ ] matrix){

  for (int column = 0; column < matrix[0].length; column++) {
      int total = 0;
      for (int row = 0; row < matrix.length; row++){

          total += matrix[row][column];
       }
          System.out.println("Sum for column " + column + " is " + total);
  }
}
```

6. (*Which row has the largest sum?*) Use variables **maxRow** and **indexOfMaxRow** to track the largest sum and index of the row. For each row, compute its sum and update **maxRow** and **indexOfMaxRow** if the new sum is greater.

```
void   findmaxRow ( int [ ][ ] matrix){
int maxRow = 0;
int indexOfMaxRow = 0;

// Get sum of the first row in maxRow
for (int column = 0; column < matrix[0].length; column++) {
     maxRow += matrix[0][column];
 }

for (int row = 1; row < matrix.length; row++) {
     int totalOfThisRow = 0;
     for (int column = 0; column < matrix[row].length; column++)
         totalOfThisRow += matrix[row][column];

          if (totalOfThisRow > maxRow) {
           maxRow = totalOfThisRow;
           indexOfMaxRow = row;
         }
   }
```

```
 System.out.println("Row " + indexOfMaxRow + " has the maximum sum of "
+ maxRow);
}
```

Ex:- write program to find sum of all elements of 2D array

```
import java.util.Scanner;

public class PassTwoDimensionalArray {
   public static void main (String[] args) {
   // Create a Scanner
  Scanner input = new Scanner(System.in);

  // Enter array values
  int [ ][ ] m = new int[3][4];
  System.out.println("EnterElements : ");
  for (int i = 0; i < m.length; i++)
       for (int j = 0; j < m[i].length; j++)
            m [ i ][ j ] = input.nextInt( );

// Display result
System.out.println("\nSum of all elements is " + sum(m) );
}

  public static int sum (int [ ] [ ] m) {
      int total = 0;
     for (int row = 0; row < m.length; row++) {
          for (int column = 0; column < m[row].length; column++) {
               total += m[row][column];
          }
      }

    return total;

}
```