

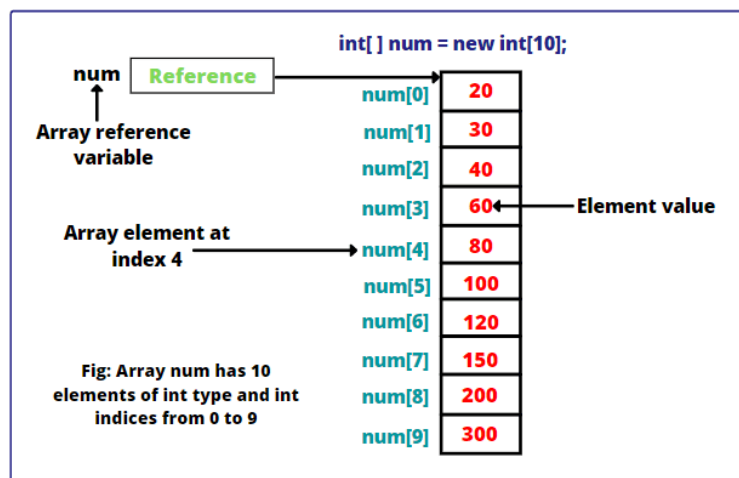


Islamic University of Gaza
Faculty of Engineering
Computer Engineering
Department



ECOM 3001 Data Structures and Algorithms [Fall 2024]

Lab "01" Topic: Arrays



Prepared by:
Eng. Waseem Alfarram

Supervised by:
Dr. Ahmad Mahdi

Lab Objectives:

- To understand the concept of the array.
- To learn how to declare, initialize, and access array elements.
- To perform basic operations on arrays such as insertion, deletion, and traversal.
- To identify types of arrays (based on dimensions) which are **one-dimensional** and **multi-dimensional** arrays.
- To implement common searching algorithms using arrays (e.g., linear searching, binary searching).
- To develop problem-solving skills using arrays.

Introduction to Arrays:

Arrays:

An **array** is a linear data structure that collects elements of the same data type and stores them in contiguous and adjacent memory locations. An array is **immutable** meaning its size cannot change without making a whole new array. Arrays work on an index system starting from 0 to (n-1), where n is the size of the array. Arrays are fundamental in computer science for organizing and manipulating data efficiently.

Syntax:

```
dataType[] arrayName;  
dataType[] arrayName = new dataType[size];
```

Array Declaration:

```
int[] numbers;
```

Array Initialization:

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

Array Combined Declaration and Initialization:

```
int[] numbers = new int[5];  
numbers[0] = 1; // Initialize using index  
int[] moreNumbers = {1, 2, 3}; // Initialize while declaring
```

Basic Operations on Arrays:

Insertion:

```
int[] arr = new int[3];  
arr[0] = 1;  
arr[1] = 2;  
arr[2] = 3;
```

Deletion: (not directly possible, but can be simulated)

```
int[] newArr = new int[arr.length - 1];  
for (int i = 0, k = 0; i < arr.length; i++) {  
    if (i == 2) { // Skip the element to delete  
        continue;  
    }  
    newArr[k++] = arr[i];  
}  
arr = newArr;
```

Traversal:

```
for (int i = 0; i < arr.length; i++) {  
    System.out.println(arr[i]);  
}
```

Types of Arrays (Based on Dimensions):

There are two types of arrays:

- One-Dimensional Arrays.
- Multi-Dimensional Arrays.

➤ One-Dimensional Arrays:

A one-dimensional array is a kind of linear array. It involves single subscripting. The [] (brackets) is used for the subscript of the array and to declare and access the elements from the array.

Syntax:

```
DataType[] ArrayName = new DataType[size];
```

For example:

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

➤ Multi-Dimensional Arrays:

In multi-dimensional arrays, we have two categories:

- Two-Dimensional Arrays.
- Three-Dimensional Arrays.

1. Two-Dimensional Arrays (Matrices):

An array involving two subscripts [] [] is known as a two-dimensional array. They are also known as the array of the array. Two-dimensional arrays are divided into rows and columns and are able to handle the data of the table.

Syntax:

```
DataType[][] ArrayName = new DataType[row_size][columnsize];
```

For example:

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

2. Three-Dimensional Arrays:

When we require to create two or more tables of the elements to declare the array elements, then in such a situation we use three-dimensional arrays.

Syntax:

```
DataType[] ArrayName = new DataType[size1][size2][size3];
```

For example:

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

Common Searching Algorithms using Arrays:

Linear Search:

```
public static int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key) {  
            return i;  
        }  
    }  
    return -1;  
}
```

Binary Search: (array must be sorted)

```
public static int binarySearch(int[] arr, int key) {  
    int low = 0;  
    int high = arr.length - 1;  
    while (low <= high) {  
        int mid = (low + high) / 2;  
        if (arr[mid] == key) {  
            return mid;  
        }  
        else if (arr[mid] < key) {  
            low = mid + 1;  
        }  
        else {  
            high = mid - 1;  
        }  
    }  
    return -1;  
}
```

Exercises:

➤ Exercise 1:

Write a java program to find the minimum and maximum element in an array and print them to the console. The result on console could be something like this:

```
Array is: {7, -2, 5, -1, 1, 8, 0, 6, 1}  
Minimum element: -2  
Maximum element: 8
```

➤ Exercise 2:

Write a java program to merge two arrays into a single array and print them to the console. The result on the console could be like this:

```
First Array: {1, 2, 3}  
Second Array: {4, 5, 6}  
Merged Array: {1, 2, 3, 4, 5, 6}
```

➤ Exercise 3:

Write a java program to remove duplicate elements from an array (**using only Array data structure**). The array should not be sorted, it should remain at the same order. The result on the console could be like this:

```
Basic Array: {6, 1, -7, 0, -4, 6, -7, 5, 0, 8}  
Array after removing duplications: {6, 1, -7, 0, -4, 5, 8}
```

Bonus Leetcode Questions:

- <https://leetcode.com/problems/check-if-two-string-arrays-are-equivalent/>
- <https://leetcode.com/problems/number-of-employees-who-met-the-target/>
- <https://leetcode.com/problems/shuffle-the-array/>
- <https://leetcode.com/problems/shuffle-string/>
- <https://leetcode.com/problems/matrix-diagonal-sum/>
- <https://leetcode.com/problems/rotate-image/>

End of Lab 😊