
ARRAYS

Aim:

To understand and implement array operations in Java.

PRE LAB EXERCISE

QUESTIONS

- ✓ What is an array?

An **array** is a collection of elements of the **same data type** stored in **contiguous memory locations** and accessed using a **single name with an index**.

- ✓ Why are arrays used?

Arrays are used to:

- Store **multiple values** of the same type under one name
- Reduce **code repetition**
- Enable **easy access** to elements using index values
- Improve **memory management** and program efficiency

- ✓ What is the difference between array and variable?

Variable

Stores only **one value**

Holds a **single memory location**

Accessed directly by name

Array

Stores **multiple values**

Uses **multiple contiguous memory locations**

Accessed using **index**

Variable

Suitable for single data

Array

Suitable for **large data set**

IN LAB EXERCISE

Objective:

To perform array operations using simple programs.

PROGRAMS:

1. Program to Read and Print Array Elements

Code:

```
import java.util.Scanner;

public class ReadPrintArray {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] arr = new int[5];
        System.out.println("Enter 5 elements:");
        for(int i = 0; i < 5; i++)
            arr[i] = sc.nextInt();
        System.out.println("Array elements are:");
        for(int i = 0; i < 5; i++)
            System.out.print(arr[i] + " ");
    }
}
```

OUTPUT:

Input:

10 20 30 40 50

Output:

Array elements are:

10 20 30 40 50

```
Enter 5 elements:  
10 20 30 40 50  
Array elements are:  
10 20 30 40 50
```

2. Program to Find Sum of Array Elements

Code:

```
import java.util.Scanner;  
  
public class SumArray {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int[] arr = new int[5];  
        int sum = 0;  
        System.out.println("Enter 5 elements:");  
        for(int i = 0; i < 5; i++)  
            arr[i] = sc.nextInt();  
        for(int i = 0; i < 5; i++)  
            sum += arr[i];  
        System.out.println("Sum = " + sum);  
    }  
}
```

OUTPUT:

Input:

5 10 15 20 25

Output:

Sum = 75

```
Enter 5 elements:
5 10 15 20 25
Sum = 75
```

3. Program to Find Largest Element in an Array

Code:

```
import java.util.Scanner;

public class LargestElement {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int[] arr = new int[5];

        System.out.println("Enter 5 elements:");

        for(int i = 0; i < 5; i++)

            arr[i] = sc.nextInt();

        int max = arr[0];

        for(int i = 1; i < 5; i++)

            if(arr[i] > max)

                max = arr[i];

        System.out.println("Largest element = " + max);

    }

}
```

OUTPUT:

Input:

12 45 23 9 30

Output:

Largest element = 45

```
Enter 5 elements:
12 45 23 9 30
Largest element = 45
```

4. Program to Reverse an Array

Code:

```
import java.util.Scanner;

public class ReverseArray {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] arr = new int[5];
        System.out.println("Enter 5 elements:");
        for(int i = 0; i < 5; i++)
            arr[i] = sc.nextInt();
        System.out.println("Reversed array:");
        for(int i = 4; i >= 0; i--)
            System.out.print(arr[i] + " ");
    }
}
```

OUTPUT:

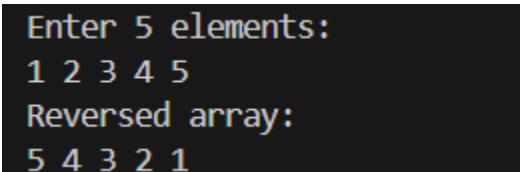
Input:

1 2 3 4 5

Output:

Reversed array:

5 4 3 2 1

A screenshot of a terminal window with a black background and light blue/green text. It shows the execution of the Java program: 'Enter 5 elements:' followed by '1 2 3 4 5' on the next line, then 'Reversed array:' followed by '5 4 3 2 1' on the next line.

```
Enter 5 elements:
1 2 3 4 5
Reversed array:
5 4 3 2 1
```

5. Program to Count Even and Odd Numbers

Code:

```
import java.util.Scanner;

public class EvenOddCount {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int[] arr = new int[5];

        int even = 0, odd = 0;

        System.out.println("Enter 5 elements:");

        for(int i = 0; i < 5; i++)

            arr[i] = sc.nextInt();

        for(int i = 0; i < 5; i++) {

            if(arr[i] % 2 == 0)

                even++;

            else

                odd++;

        }

        System.out.println("Even = " + even);

        System.out.println("Odd = " + odd);

    }

}
```

OUTPUT:

Input:

2 7 4 9 10

Output:

Even = 3

Odd = 2

```
Enter 5 elements:  
2 7 4 9 10  
Even = 3  
Odd = 2
```

6. Program to Sort Array in Ascending Order

Code:

```
import java.util.Scanner;  
  
public class SortArray {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int[] arr = new int[5];  
  
        int temp;  
  
        System.out.println("Enter 5 elements:");  
  
        for(int i = 0; i < 5; i++)  
            arr[i] = sc.nextInt();  
  
        for(int i = 0; i < 5; i++) {  
            for(int j = i + 1; j < 5; j++) {  
                if(arr[i] > arr[j]) {  
                    temp = arr[i];  
                    arr[i] = arr[j];  
                    arr[j] = temp;  
                }  
            }  
        }  
  
        System.out.println("Sorted array:");  
  
        for(int i = 0; i < 5; i++)  
            System.out.print(arr[i] + " ");  
    }  
}
```

```
}
```

OUTPUT:

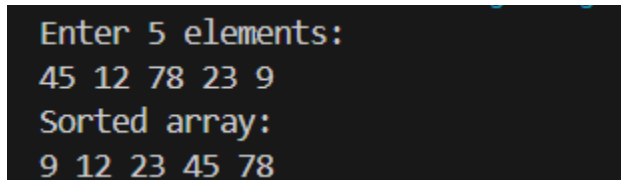
Input:

45 12 78 23 9

Output:

Sorted array:

9 12 23 45 78



```
Enter 5 elements:
45 12 78 23 9
Sorted array:
9 12 23 45 78
```

7. Program to Find Second Largest Element

Code:

```
import java.util.Scanner;

public class SecondLargest {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] arr = new int[5];

        System.out.println("Enter 5 elements:");
        for(int i = 0; i < 5; i++)
            arr[i] = sc.nextInt();

        int largest = arr[0];
        int second = arr[0];
        for(int i = 0; i < 5; i++) {
            if(arr[i] > largest) {
                second = largest;
                largest = arr[i];
            }
        }
    }
}
```



```

    }
}
System.out.println("Second largest = " + second);
}
}

```

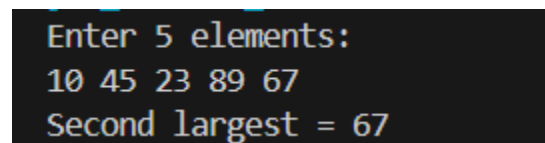
OUTPUT:

Input:

10 45 23 89 67

Output:

Second largest = 67



```

Enter 5 elements:
10 45 23 89 67
Second largest = 67

```

8. Program for Matrix Addition (2D Array)

Code:

```

import java.util.Scanner;

public class MatrixAddition {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int[][] a = new int[2][2];
        int[][] b = new int[2][2];
        int[][] sum = new int[2][2];

        System.out.println("Enter elements of matrix A:");
        for(int i = 0; i < 2; i++)
            for(int j = 0; j < 2; j++)
                a[i][j] = sc.nextInt();

        System.out.println("Enter elements of matrix B:");
        for(int i = 0; i < 2; i++)

```

```
        for(int j = 0; j < 2; j++)
            b[i][j] = sc.nextInt();
    for(int i = 0; i < 2; i++)
        for(int j = 0; j < 2; j++)
            sum[i][j] = a[i][j] + b[i][j];
    System.out.println("Sum matrix:");
    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 2; j++)
            System.out.print(sum[i][j] + " ");
        System.out.println();
    }
}
```

OUTPUT:

Matrix A:

1 2

3 4

Matrix B:

5 6

7 8

Sum matrix:

6 8

10 12

```
Enter elements of matrix A:
1 2
3 4
Enter elements of matrix B:
5 6
7 8
Sum matrix:
6 8
10 12
```

POST LAB EXERCISE

- ✓ Why is array indexing usually started from zero instead of one?

Array indexing starts from **zero** because the index represents the **offset from the base memory address**.

- The first element is at offset **0**, so it is accessed as index 0
- This makes address calculation faster and more efficient

- ✓ What happens if we try to access an array element outside its declared size?

Accessing an element outside the declared size leads to **undefined behavior**.

- It may cause **runtime errors**
- It can produce **garbage values**
- In some languages, it may cause **program crash**

- ✓ How does memory allocation differ for static arrays and dynamic arrays?

Static Array

Memory allocated at **compile time**

Fixed size

Stored in **stack memory**

Faster access

Dynamic Array

Memory allocated at
runtime

Size can be **changed**

Stored in **heap memory**

Slightly slower due to
runtime allocation

- ✓ Why is searching faster in arrays compared to linked lists?

Searching is faster in arrays because:

- Arrays support **direct access** using index
- Elements are stored in **contiguous memory**
- Binary search can be applied (if sorted)

In linked lists, traversal must start from the **first node**, making searching slower.

- ✓ What is the difference between contiguous and non-contiguous memory allocation?

Contiguous Memory Allocation

Memory locations are **adjacent**

Faster access

Non-Contiguous Memory Allocation

Memory locations
are **scattered**

Slower access

Contiguous Memory Allocation

Used in **arrays**

Less flexible

Non-Contiguous Memory Allocation

Used in **linked lists**

More flexible

Result:

Thus the array operations were executed successfully.

ASSESSMENT

Description	Max Marks	Marks Awarded
Pre Lab Exercise	5	
In Lab Exercise	10	
Post Lab Exercise	5	
Viva	10	
Total	30	
Faculty Signature		