Arrays – Introduction to Arrays

1. What is an Array?

An **array** is a linear data structure used to store **multiple elements of the same data type** in **contiguous memory locations**.

Each element in an array is accessed using an **index**, which represents its position.

In simple terms, an array allows us to store many values under **one variable name** and access them efficiently.

---

2. Why Do We Need Arrays?

Without arrays:

- We need multiple variables to store similar data
- Code becomes repetitive and difficult to manage

With arrays:

- Data is stored in an organized manner
- Operations like searching, sorting, and traversal become easier
- Memory access becomes faster due to contiguous storage

Example scenario:

- Storing marks of 50 students
- Without array → 50 variables
- With array → 1 array of size 50

---

3. Characteristics of Arrays

Key characteristics of arrays include:

1. **Fixed Size**

   The size of an array is decided at the time of creation and cannot be changed later.

2. **Homogeneous Elements**

   All elements stored in an array must be of the same data type (e.g., all integers).

3. **Index-Based Access**

   Each element is accessed using an index value starting from **0**.

4. **Contiguous Memory Allocation**

   All elements are stored in adjacent memory locations.

5. **Fast Access Time**

   Any element can be accessed in constant time using its index.

---

4. Array Indexing

- Index starts from **0**
- Last index = `size − 1`

Example (Array of size 5):

Index	0	1	2	3	4
Value	10	20	30	40	50

Accessing:

- First element → index `0`
- Last element → index `4`

---

5. Memory Representation of Arrays

Arrays are stored in **contiguous memory blocks**.

If:

- Base address = 1000
- Each integer takes 4 bytes

Then:

- `arr[0]` → 1000
- `arr[1]` → 1004
- `arr[2]` → 1008

This structure allows **direct calculation of address** using the index.

---

6. Types of Arrays

1️⃣ One-Dimensional Array

- Stores elements in a single line
- Example: list of numbers

2️⃣ Two-Dimensional Array

- Stored in rows and columns
- Example: matrix

3️⃣ Multi-Dimensional Array

- More than two dimensions
- Used in advanced applications

For beginners, **one-dimensional arrays** are most commonly used.

---

7. Basic Operations on Arrays

Even at an introductory level, arrays support these operations:

- Traversal (visiting each element)
- Insertion
- Deletion
- Searching
- Updating values

These operations form the foundation for advanced algorithms.

---

8. Advantages of Arrays

- Easy to store and manage large amounts of data
- Fast access using index
- Useful for implementing other data structures
- Simple structure, easy to understand

---

9. Limitations of Arrays

- Fixed size (cannot grow dynamically)
- Insertion and deletion are costly
- Memory wastage if size is not used fully
- Cannot store mixed data types

---

10. Real-World Examples of Arrays

- Student marks list
- Daily temperature readings
- Scores in a game
- Monthly sales data
- Sensor readings in IoT devices

---

11. Summary

- Arrays store multiple elements of the same type
- They use index-based access
- Memory is allocated contiguously
- Arrays are the foundation of most DSA concepts
- Understanding arrays is essential before learning searching, sorting, and other data structures

---