**How Arrays are Represented in Memory**

* Arrays are stored in **contiguous memory locations**.
* Each element is of the **same data type**, and the index of the first element is 0.
* The memory address of any element can be calculated using:

nginx

Copy code

Address of arr[i] = Base Address + (i × size\_of\_datatype)

**Advantages**

* **Direct Access**: Constant-time access using index arr[i].
* **Cache-friendly**: Due to contiguous memory, arrays work well with CPU caching.
* **Simple structure**: Easy to use and implement.

**2. Time Complexity Analysis**

| **Operation** | **Time Complexity** | **Explanation** |
| --- | --- | --- |
| Access | O(1) | Direct index-based access |
| Search | O(n) (unsorted)O(log n) (sorted & binary search) | Linear or binary search depending on data |
| Traverse | O(n) | Visiting all elements |
| Insert (end) | O(1) (if space available) | Insert at last index |
| Insert (middle/start) | O(n) | Needs shifting elements |
| Delete | O(n) | Shifting needed to fill gap |

**Limitations of Arrays**

| Limitation | Details |
| --- | --- |
| Fixed Size | Size must be defined at declaration; can't grow dynamically. |
| Costly Insertion/Deletion | Shifting elements causes overhead. |
| Wasted Space | If over-allocated for future use. |
| Same Type Only | Stores only one data type (unless Object array). |

**When to Use Arrays**

Use arrays when:

* Size is known and fixed.
* You need fast random access.
* Insertions and deletions are rare.
* Performance is critical and memory is tightly managed.