**Employee Management System**

**Arrays are Represented in Memory** :

Arrays are a collection of elements stored in contiguous memory locations. This allows for efficient indexing, as each element can be accessed directly by calculating its address using the base address and the element's index.

**Advantages of Arrays**:

1. Constant-Time Access: Elements can be accessed in O(1) time using their index.

2. Cache-Friendly: Contiguous memory allocation helps in better cache utilization, leading to faster access.

3. Simple Implementation: Arrays are easy to implement and use, making them a straightforward choice for many applications.

**Analysis**

1. Add Employee - Time Complexity: O(1) (Amortized, if we ignore the array resizing when it is full)

2. Search Employee- Time Complexity: O(n)

3. Traverse Employees- Time Complexity: O(n)

4. Delete Employee- Time Complexity: O(n) (Due to the need to shift elements after deletion)

**Limitations of Arrays and When to Use Them** :

1. Fixed Size: Arrays have a fixed size, which means we need to define the maximum capacity upfront. This can lead to either wasted space or insufficient capacity.

2. Insertion and Deletion: While arrays offer constant-time access, insertion (at any position other than the end) and deletion require shifting elements, which is O(n) in time complexity.

3. Dynamic Size: For applications requiring dynamic resizing, dynamic arrays (like `ArrayList` in Java) or other data structures (like linked lists) might be more suitable.

**When to Use Arrays**:

- When the number of elements is known in advance and is relatively small.

- When constant-time access by index is crucial.

- When the overhead of resizing or managing more complex data structures is not justified.

In conclusion, arrays are useful for simple, fixed-size collections with frequent access by index. For more flexible and scalable solutions, consider dynamic arrays or other data structures.