**Employee Management System**

**Array Representation in Memory:**

* **Contiguous allocation:** Arrays occupy a continuous block of memory.
* **Fixed-size:** The size is determined at creation and remains constant.
* **Index-based:** Elements are accessed using numerical indices. They allow for O(1) time complexity for accessing any element by its index.
* **Memory Organization:** 
  + Starting address (base address) of the array is stored
  + Each element occupies the same amount of memory
  + Address of any element = Base address + (Index \* Size of each element)

**Advantages of Arrays:**

1. **Fast access(O(1) time complexity) :**Direct access to any element using its index.
2. **Memory efficiency**: No additional overhead for storing relationships between elements
3. **Cache friendliness:** Contiguous memory allocation improves cache performance
4. **Simplicity:** Arrays are easy to understand and implement

**Time Complexity Analysis**

* **Add Employee**: O(1) - Efficient if there is space in the array.
* **Search Employee**: O(n) - Linear search requires examining potentially all elements.
* **Traverse Employees**: O(n) - Necessitates iterating over all elements.
* **Delete Employee**: O(n) - Requires both a search and shifting of elements.

**Limitations of Array :**

* **Fixed Size:** Unable to resize after creation, leading to potential waste or shortage of space.
* **Inefficient Insertions/Deletions:** Adding or removing elements requires shifting, which can be slow.
* **Memory Overhead:** Overestimating size wastes memory; underestimating requires costly resizing.
* **Absence of Dynamic Sizing:** Cannot grow or shrink dynamically without creating new arrays.

**When to Use Arrays:**

* **Simple, Small Datasets**: For straightforward cases with minimal overhead.
* **Memory-Constrained Environments**: When memory overhead needs to be minimal.