**Exercise 4: Employee Management System**

**Array Representation in Memory:**

* **Contiguous Memory Allocation:** Arrays are stored in contiguous memory locations, which means each element is located next to its neighbour.
* **Indexing:** Elements in an array are indexed, usually starting from 0. This allows for O(1) time complexity for accessing elements using their index.
* **Fixed Size:** The size of an array is fixed at the time of creation and cannot be changed.

**Advantages of Arrays:**

* **Fast Access:** Direct access to any element using its index with O(1) time complexity.
* **Memory Efficiency:** Arrays have low overhead since they are stored contiguously in memory.
* **Ease of Use:** Simple to declare and use, especially for homogeneous data types.

**Time Complexity Analysis:**

* **Add Operation:**
  + **Time Complexity:** O(1) (assuming there is space available in the array)
  + **Explanation:** Adding an element to the end of the array is a constant time operation.
* **Search Operation:**
  + **Time Complexity:** O(n)
  + **Explanation:** In the worst case, the search operation might need to check each element in the array.
* **Traverse Operation:**
  + **Time Complexity:** O(n)
  + **Explanation:** Traversing through all elements in the array requires visiting each element once.
* **Delete Operation:**
  + **Time Complexity:** O(n)
  + **Explanation:** In the worst case, the deletion operation might need to shift all elements after the deleted element to fill the gap.

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

**Limitations:**

* **Fixed Size:** Arrays have a fixed size, which means the maximum number of elements must be known in advance.
* **Inefficient Deletions/Insertions:** Deleting or inserting elements in the middle of the array requires shifting elements, leading to O(n) time complexity.
* **Memory Wastage:** If the array's capacity is overestimated, it can lead to unused memory spaces.

**When to Use Arrays:**

* **When the Size is Known:** Arrays are suitable when the number of elements is known and fixed.
* **Fast Access Required:** Arrays are efficient for applications requiring fast random access to elements.
* **Homogeneous Data:** Arrays are ideal for storing a collection of elements of the same data type.