**Exercise 4: Employee Management System**

**Scenario:**

You are developing an employee management system for a company. Efficiently managing employee records is crucial.

**Steps:**

1. **Understand Array Representation:**
   * Explain how arrays are represented in memory and their advantages.

**Ans.** We can represent an array in various ways in different programming languages. Lets understand with an example:

For example, if we have an array **arr** with 5 elements, the memory representation might look like this:

int array[5] = {1,2,3,4,5}

the memory representation be like

A diagram of a number

Description automatically generated

* In array ndex starts with 0.
* The array's length is 10, which means we can store 10 elements.
* Each element in the array can be accessed via its index.

**Advantages of arrays:**

* **Efficient memory usage**: Arrays store elements in contiguous memory locations, which makes them memory-efficient.
* **Fast access time**: Arrays provide fast access to elements using their index, which makes them suitable for applications that require frequent access to elements.
* **Simple implementation**: Arrays are easy to implement and understand, making them a popular choice for many applications.

1. **Analysis:**
   * Analyze the time complexity of each operation (add, search, traverse, delete).

Time Complexity: In both Average case and Worst case time complexity are same for array

* + - **Add :** O(1), because we only need to increment the size variable and assign the new employee to the next available slot in the array.
    - **Search:** O(n), because we need to iterate through the entire array to find the employee with the given ID.
    - **Traverse:** O(n), because we need to iterate through the entire array to print all employees.
    - **Delete**: O(n), because we need to iterate through the entire array to find the employee with the given ID and then shift elements to the left to fill the gap.
  + Discuss the limitations of arrays and when to use them.

**Limitation:**

* **Homogenous:** Array is homogenous. It means that the elements with similar data type can be stored in it.
* **Fixed size: In array:** there is static memory allocation that is size of an array cannot be altered.
* **Memory wastage:** There will be wastage of memory if we store less number of elements than the declared size.

**When we can use:**

* **Small datasets**: Arrays are suitable for small datasets where the size is fixed and known in advance.
* **Frequent access**: Arrays are suitable for applications that require frequent access to elements, such as caching or buffering.
* **Simple implementation**: Arrays are suitable for applications where simplicity and ease of implementation are more important than performance.