**4\_Employee Management System**

**a. How Arrays Are Stored**

* Think of an array like a row of mailboxes placed side by side. Each box (element) is placed right next to the other in memory, one after the other, this is called **contiguous memory**.
* Because of this, if you know where the first element is, the computer can instantly calculate where the 5th or 100th one is, just like jumping ahead to mailbox #5 without opening the earlier ones.

**Why This Is Useful**

|  |  |
| --- | --- |
| **Feature** | **Why It’s Helpful** |
| **Instant Access** | You can quickly jump to any element using its index, it's like teleporting directly to it! This takes constant time: **O(1)**. |
| **Compact & Clean** | Since everything is packed together, no extra memory is wasted on links or pointers, it's efficient. |
| **Predictable Order** | Because the layout is consistent, arrays play well with CPU memory caches, making them **faster in practice** for many operations. |

**Time Complexity**

|  |  |  |
| --- | --- | --- |
| **Operation** | **Time** | **Meaning** |
| Add | O(1) | Fast, just add at the end. |
| Search | O(n) | May need to check each item. |
| Traverse | O(n) | Visit each employee one by one. |
| Delete | O(n) | Need to shift items after deletion. |

**Limitations of Arrays**

|  |  |
| --- | --- |
| **Limitation** | **Why It’s a Problem** |
| Fixed Size | Can’t grow or shrink easily. |
| Costly Deletion | Shifting elements takes time. |
| Wasted Space | Empty slots waste memory. |

**Use Arrays When**

* You know the size in advance.
* Need fast access by index.
* Want a simple and fast structure.

Otherwise, use **ArrayList**, **HashMap**, or **LinkedList** for more flexibility.