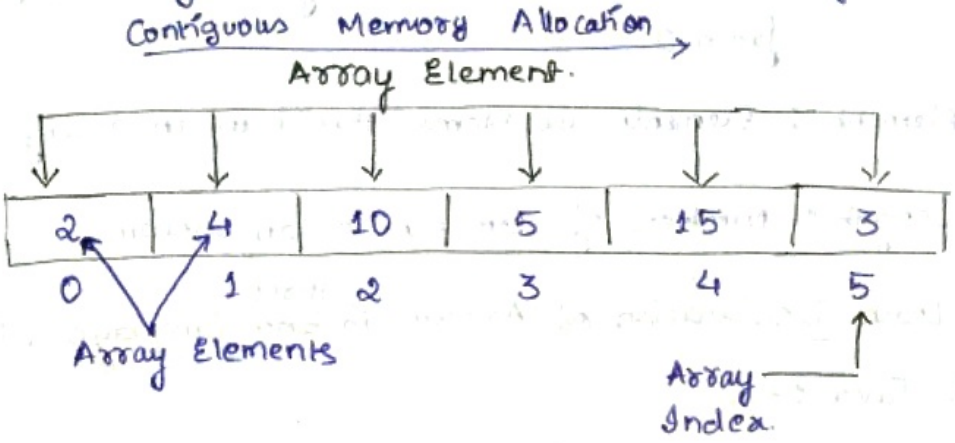
DAY - 2

#Days_Until_I_get_Placed.

* ARRAY DATA STRUCTURE :- (Sequential / Linear Data Structure)
  - It stores a collection of elements in a contiguous block of memory.
  - For accessing elements we use indices. (of same data type)



  - In array we start indexing from 0 as it's initial element

* Need for Array ??
  - storing data.
  - Implementing data structures as stacks and Queues
  - Representing data in tables and matrices
  - Creating dynamic data structure such as linked lists and trees.

ARRAYS (two types)

One Dimensional array.
(store a single row of elements).

Two dimensional array
(store Multiple rows of element).

ARRAY OPERATIONS.

Traversal
Visiting each element of array in specific order.

Insertion
Adding new element to an array at specific index

Deletion
Removing an element from an array at specific index.

Searching
finding index of an element in an array.

★ the Array has a fixed sized meaning once the size is given to it, it cannot be changed., we can't shrink it nor we can expand it.

★ Basic terminologies of Array :-

- Array Index : elements are identified by their indexes, starts from 0.
- Array Element : Elements are items stored in an array.
- Array Length : Number of elements it can contain.

★ we will Learn Declaration of Array in ~~two~~ three languages, Python C++ and Java :-

1) C++ Code :-

```
int arr[5];     // integer type
char arr[10];   // character / string type
float arr[20];  // float type
```

2) Java code :-

```
int arr[];     // integer type
char arr[];    // character type
float arr[];   // float type
```

3) Python3 code :-

```
import array
arr = array.array('i')     // integer type
arr = array.array('b')     // character type
arr = array.array('f')     // float type
```

* Initialization of Array :-

1) C++ code :-

```
int arr[] = {1,2,3,4,5};
char arr[5] = {'a','b','c','d','e'};
float arr[5] = {1.4, 2.0, 24, 5.0, 0.0};
```

2) Java Code :-

```
int arr[] = {1,2,3,4,5};
char arr[] = {'a','b','c','d','e'};
float arr[] = {1.4f, 2.0f, 24f, 5.0f, 0.0f};
```

3) Python Code :-

```
import array
arr = array.array('i', [1,2,3,4,5])
arr = array.array('b', [])
arr = array.array('f')
```

* NB :- The idea of an array is to represent many instances in one variable.

```
int v1 = 10;
int v2 = 20;
int v3 = 30;
:
```

Multiple variables to store each value.

| 10 | 20 | 30 | 40 | ... |

Single Array to store all values

Types of Arrays

- On basis of memory allocation
  - Static
  - Dynamic
- On the basis of Dimensions
  - One-Dimensional
  - Multi Dimensional
    - Two dimensional array
    - Three Dimensional array
    - And so on.

★ On the basis of memory allocation :-

1. Static Arrays :- • Memory is allocated at compile time having a fixed size of it, cannot be updated later.
   • Also known as static or compile time memory allocation.

eg :-
```
// Static Integer array in C++
int arr[5] = {1,2,3,4,5};

// static array in Java.
int[] arr = {1,2,3,4,5};

// Static array in Python
arr = array.array('i',[1,2,3,4,5])
```

2. Dynamic Arrays :- • memory is allocated at run time but not having a fixed size.
   • Also known as run-time or dynamic memory allocation.

eg :-
```
// Dynamic Array in C++
int* arr = new int[5];

// In Java
ArrayList<Integer> arr = new ArrayList<>();

// Dynamic Array in Python
arr = []
```

★ On the basis of Dimensions :-

1. One - Dimensional Array (1-D Array) :- Elements are stored one after another.

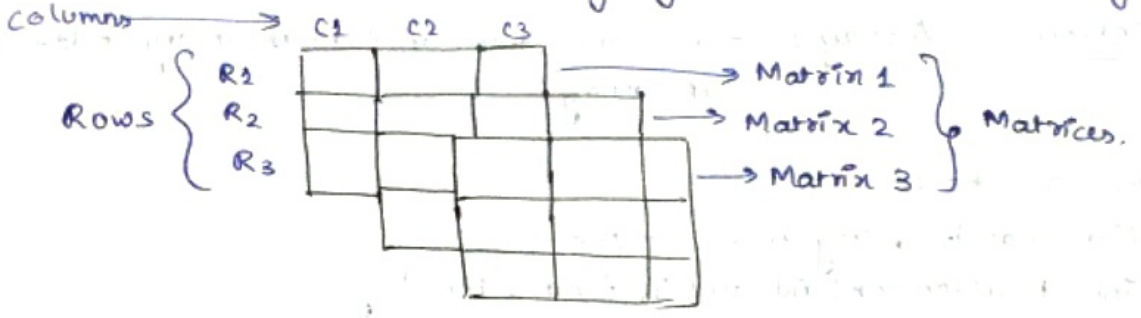
Index →	0	1	2	3	4
Element →	5	10	20	25	30

2. Multi - Dimensional Array (2-D Array) :- An array of arrays or as a matrix consisting of rows and columns.

Rows ↓ / Columns →

	0	1	2
0	a_{00}	a_{01}	a_{02}
1	a_{10}	a_{11}	a_{12}
2	a_{20}	a_{21}	a_{22}

3. Three Dimensional Array :- contains three dimensions, can be considered as an array of two dimensional arrays



★ Operations on Array :-

1. Array Traversal :- Visiting all the elements of array once.

Codeing in C++ :-

```
int arr[] = {1,2,3,4,5};
int len = sizeof(arr) / sizeof(arr[0]);
// Traversing over arr[]
for (int i=0 ; i<len ; i++) {
    cout << arr[i] << " ";
```

Code in Java :-

```
int arr[] = {1,2,3,4,5};
for (int i = 0 ; i<arr.length ; i++) {
    System.out.println (arr[i] + " ");
```

Time complexity :-
Best = $\Omega(N)$
Average = $\theta(N)$
Worst = $O(N)$

Space Complexity :-
Best = $\Omega(1)$
Average = $\theta(1)$
Worst = $O(1)$

2. Insertion in Array :- Inserting one or multiple elements at any position in array.

Code in C++ :-

```
void insertElement(int arr[], int n, int x, int pos)
{
    for(int i = n-1 ; i>=pos ; i--)
        arr[i+1] = arr[i];
    arr[pos] = x;
}
```

Code in Java :-

```
static void insertElement (int arr[], int n, int x, int pos)
{
    // shift elements to right and add
    for (int i = n-1 ; i>=pos ; i--)
        arr[ i+1 ] = arr[i];
```

Time Complexity :-
Best = $\Omega(1)$
Average = $\theta(N)$
Worst = $O(N)$

Space Complexity :-
Best = $\Omega(1)$
Average = $\theta(N)$
Worst = $O(N)$

```
        arr[pos] = x;
    }
```

3) Deletion in Array :- - We can delete an element at any index in an array:

Code in c++ :-

```
// To search a key to be deleted
int findElement(int arr[], int n, int key);

int deleteElement(int arr[], int n, int key)
{
    int pos = findElement(arr, n, key);
    if (pos == -1) {
        cout << "Element not found";
        return n;
    }
    // Deleting element
    int i;
    for (i = pos; i < n - 1; i++)
        arr[i] = arr[i + 1];
    return n - 1;
}
int findElement(int arr[], int n, int key)
{
    int i;
    for (i = 0; i < n; i++)
        if (arr[i] == key)
            return i;
    return -1;
}
```

Time Complexity :-
Best - $\Omega(1)$
Average - $\theta(N)$
Worst - $O(N)$

Space Complexity :-
Best - $\Omega(1)$
Average - $\theta(N)$
Worst - $O(N)$

Code in Java :

```
static int findElement(int arr[], int n, int key)
{
    int i;
    for (i = 0; i < n; i++)
        if (arr[i] == key)
            return i;
    return -1;
}
```

```
static int deleteElement (int arr[], int n, int key)
{
    int pos = findElement (arr, n, key);
    if (pos == -1) {
        System.out.println ("Element not found");
        return n;
    }
    int i;
    for (i = pos; i<n-1; i++)
        arr[i] = arr[i+1];

    return n-1;
}
```

4) Searching in Array :- - we can traverse over an array and search for an element.

Code in C++ :

```
int findElement (int arr[], int n, int key)
{
    int i;
    for (i=0; i<n; i++)
        if (arr[i] == key)
            return i;

    return -1;
}
```

Time Complexity :-

Best = $\Omega(1)$

Average = $\Theta(N)$

Worst = $O(N)$

Space complexity :-

Best = $\Omega(1)$

Average = $\Theta(1)$

Worst = $O(1)$

Code in Java :-

```
int findElement (int arr[], int n, int key)
{
    for (int i = 0; i<n; i++)
        if (arr[i] == key)
            return i;

    return -1;
}
```

★ Applications →

- Storing and accessing data, Sorting, Searching, Matrices, stacks and queues, Graphs, and Dynamic Programming.

★ Subarray → A subarray is a contiguous part of array i.e subarray is an array that is inside another array.

- In general, for an array of size n, there are $n*(n+1)/2$ non empty subarrays.

- eg :- array = [1, 2, 3, 4]

subarrays : (1), (2), (3), (4), (1,2), (2,3), (3,4), (1,2,3) (2,3,4) and (1,2,3,4).

★ Subsequence → A subsequence is a sentence that can be derived from another sequence by removing zero or more elements, without changing the order of the remaining elements.

- eg :- array = [1,2,3,4]

subsequence = (1), (2), (3), (4), (1,2), (1,3), (1,4), (2,3) (2,4) .... 15 subsequences.

- In general, we can have $(2^n - 1)$ non-empty sub-sequences in total.

★ Subset → If a set has all its elements belonging to other sets, this set will be known as a subset of other set.

- represented as $A \subseteq B$

- eg :- let Set_A = {m, n, o, p, q}

Set_B = {k, l, m, n, o, p, q, r}

Then $A \subseteq B$ = {m, n, o, p, q}

Then A is subset of B.