ARRAYS

WEEK 3
ARRAY - L1

Array :

→ Data structure to store similar items.
(i.e of same datatype)

→ Continuous memory locations.

→ use case

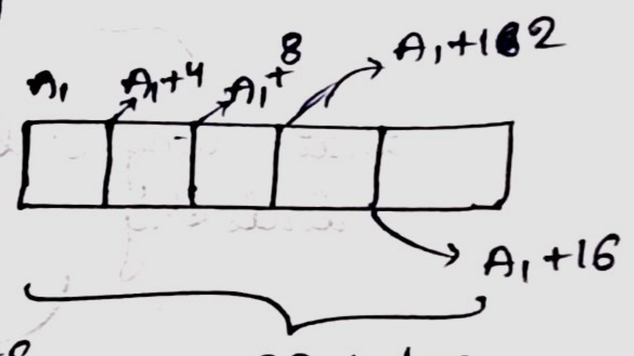
↳ for multiple same kind of data.

Continuous memory allocation

↳ memory wastage if required amount of memory locations are present, but not in continuous way.

Eg : int a = 5 ; → [5] (a) | Char arr [10];

Declaration : int arr [5]; →



A1 ⇒ base address
Int → 4 bytes

cout << arr ; → A1 → address of first index of array
(address of integer)

cout << &arr ; → A1 → base address of array.
(address of array)

Initialisation :

int arr [ ] = {2, 4, 6, 8};

int arr [5] = {1, 3, 5, 7, 9};

int arr [10] = {6, 9, 0, 7}; → remaining 6 will be zero.

int arr [4] = {2, 16, 12, 7, 9, 33}; → ERROR!

int arr [8] = {0}; → Initializing all values with 0.

* Making array at runtime:

```
int n;
cin >> n;
int array[n]; ⟶ BAD PRACTICE
```

Index and Access in memory

```
int arr[5] = {10, 20, 30};
```

↳ 0th based indexing
⇒ 0 to (n-1)

arr

10	20	30	0	0
0	1	2	3	4

arr[2] = 30;

★ arr[i] ⟶ value at address [A_1 + (i * size of datatype)]

- BA
- index
- ↳ eg: for int it's 4.

↳ that's why 0 based indexing

↳ due to Internal working.

Arrays and Function :-

↳

```
func(int arr[], int size)
{
-----
}
```

↳ array is passed by reference because pointer is passed by value.

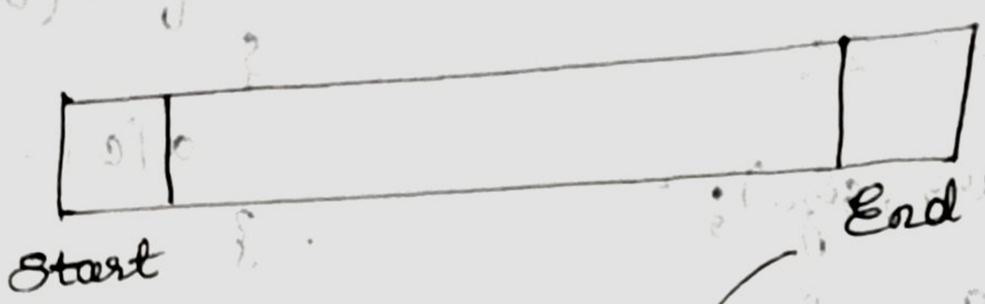
↳ updation in actual array.

↳ always pass size alongwith array.

NOTE:

arr here is not an array, it is pointer pointing to first index of array.
[Will learn ahead]

2 - Pointer Approach :-



use of two variables as extreme points.

i/p :- int arr[] = {10,20,30,40,50,60,70};

o/p :- 10 70 20 60 30 50 40

Code:

```
int arr[] = {10,20,30,40,50,60,70};
int size = 7;
int start = 0;
int end = size-1;

// or use condition start <= end
while (true) {
    if (start > end)
        break;
    if (start == end)
        cout << arr[start] << "__";
    else
        cout << arr[start] << "__";
        cout << arr[end] << "__";
    start++;
    end--;
```

To find size of an Array :

```
int arr [ ] = {1, 2, 3, 4};
int size = size of (arr) / size of (int);
                              ↳ datatype
```

Q: Reverse an Array.

i/p : int arr [5] = {1, 2, 3, 4, 5};

o/p : 5 4 3 2 1

Code :

```
int arr [5] = {1, 2, 3, 4, 5};
int start = 0;
int end = 4;

while (start <= end)
  {
    swap (arr [start] , arr [end ] ) ;
    start ++ ;
    end -- ;
  }

for (int i=0; i < 5 ; i++)
  {
    cout << arr [i] << "__";
  }
```