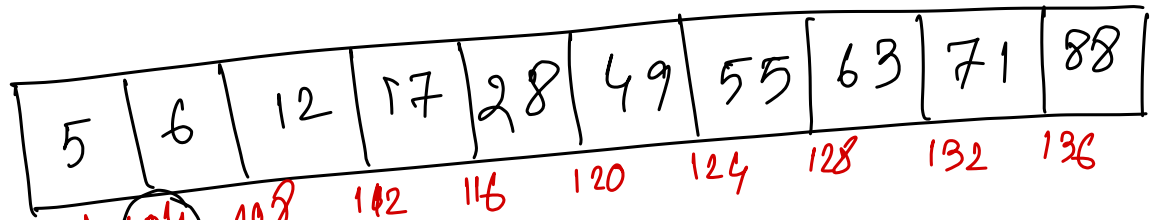
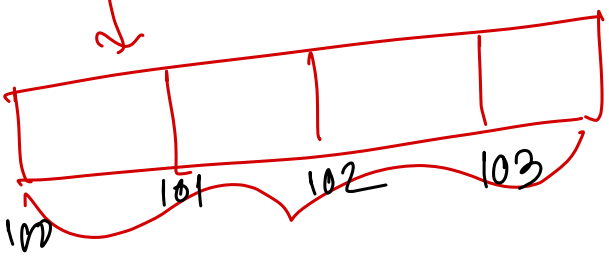
Data Structures

- Linear
  - Static
    - Array
  - Dynamic
    - Linked List
    - Stack
    - Queue
- Non-Linear
  - Tree
  - Graph
- Hash Table
  - Hashing

Array

A collection of Homogenious data types in a sequential manner in contiguous memory location.





ECE dept. (NITA)

2nd year ⇝ 125

Data Structures

125 → variable

int marks[125];

- int → datatype
- marks → Array name
- 125 → Size of the array.

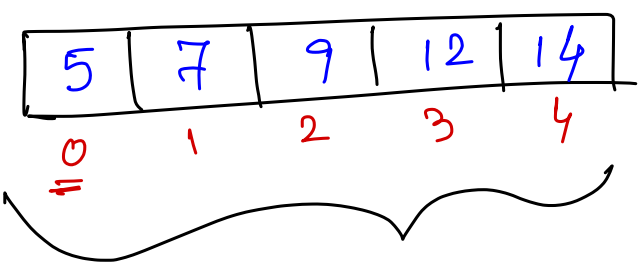
Array
- 1-Dimentional
- 2-Dimentional.
- 3-Dimentional.
- multi Dimentional

1-Dimentional Array

int arr[5];

arr is an array of 5 integers.

int arr[5]



1st element = arr[0].
2nd " = arr[1]
3rd " = arr[2]
4th " = arr[3]
5th " = arr[4]

last element
arr[size-1]

[ ] → Subscript Operator

* used to extract the element from any position of the array.

arr[i++]

arr[i+j]

arr[2*i]

arr[j--]

i = —
j = —

size

```
int main() {
    int arr[5];
    printf("Enter the array :- ");
    for(int i=0; i<5; i++){
        scanf("%d", &arr[i]);
    }
    printf("The Array is:- ");
    for(int i=0; i<n; i++){
        printf("%d\t", arr[i]);
    }
```

input

};

Initialization of Array.

int arr[] = { 2, 3, 9, 5, 6 }

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

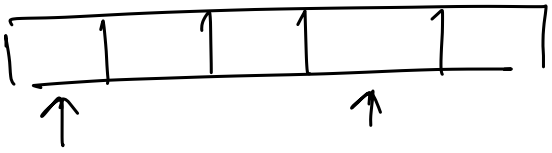
int b[5] = { 2, 3, 4 };

b[3] b[4]

↓

{ 2, 3, 4, 0, 0 }

* The most advantageous thing of an array is having the direct access of the elements.



marks[125]

marks[92] = 80

int arr[5] = { 11, 12, 13, 14, 15, 16, 17, 18 }

error

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

int b[5];

b = a → Error

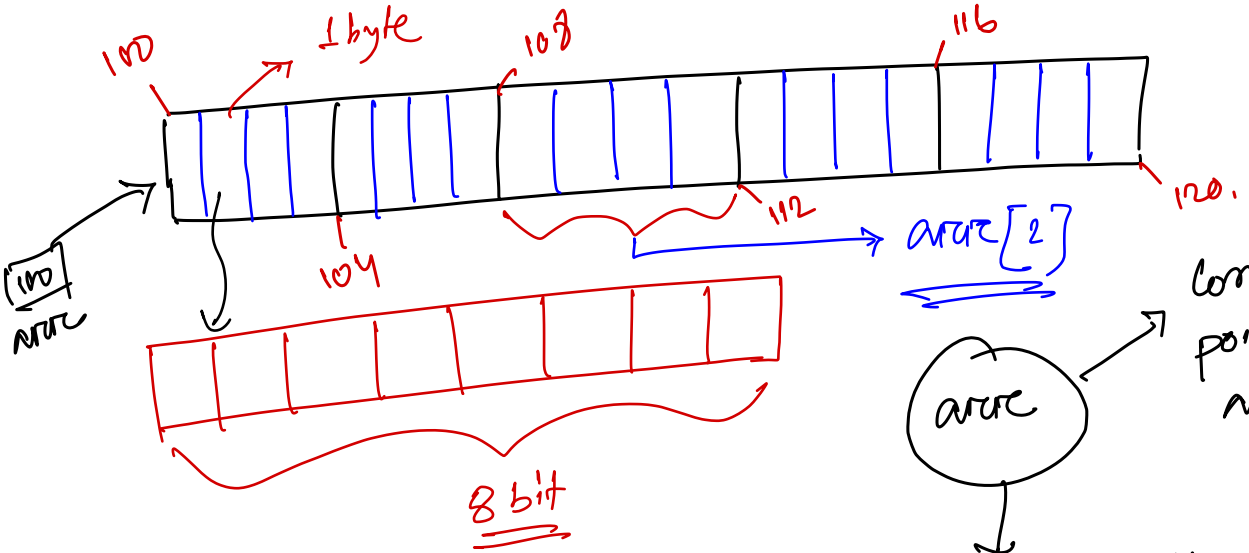
} won't be copied from a to b

```
for(int i=0; i<5; i++){
    b[i] = a[i];
}
```

Copy in this way.

(*) Array itself a pointer

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



arr → constant pointer pointing to the array.

arr → contain the base address of the array.

arr[0] → *(arr)
arr[1] → *(arr + 1)
arr[2] → *(arr + 2)
arr[3] → *(arr + 3)
arr[4] → *(arr + 4)
arr[5] → *(arr + 5)
arr[i] → *(arr + i)

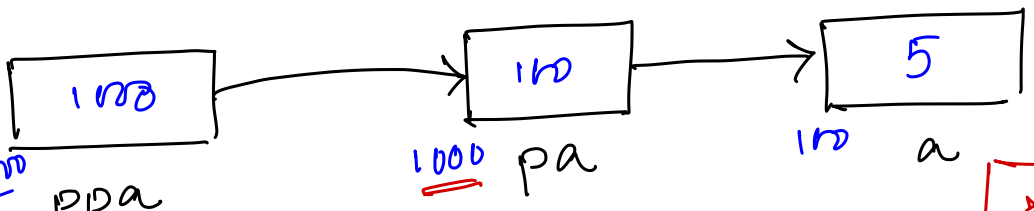
arr + i = i + arr

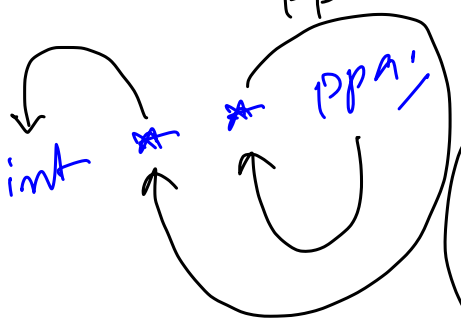
*(arr + i) = *(i + arr)

arr[i] = i[arr]

Same

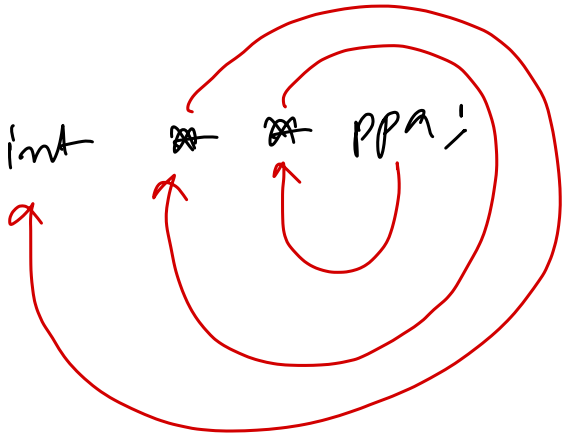
Pointers to pointers






```
int a = 5;
int *p = &a;
int **ppa = &p;
```

**ppa
*(*ppa)
*pa
*(&a)
a;



⇒ ppa is a pointer which is pointing to another pointer which points to an integer

	a	pa	ppa	5
Value of a	a	*pa	**ppa	5
Address of a	&a	pa	*ppa	100
Value of Pa	&a	pa	*ppa	100
Address of pa	N/A	&pa	ppa	1000
value of ppa	N/A	&pa	ppa	1000
Address of ppa	N/A	N/A	&ppa	2000