Ch-6(L-2) : Pointers II

Pointers in Array

▼ int arr

12	44	66	18	46

- & of any index will give address of it. Ex - `&arr[1]`

▼ Variable of array(`arr`) , `&arr` , `&arr[0]` all the three denotes the base address of an array

```
int arr[10] = {12,44,66,18,46};
int* ptr = arr;
cout << arr; //output -> base add of arr
cout << &arr; //output -> base add of arr
cout << &arr[0]; //output -> base add of arr
```

▼ See the following results, `(*arr)+1` → ans will be 13, `*(arr+1)` → ans will be 44

```
int arr[10] = {12,44,66,18,46};
int* ptr = arr;
cout << (*arr)+1; //output -> 13
cout << *(arr+1); //output -> 44
cout << *(arr+5); //output -> garbage value, segmentation
```

 Internally `arr[i] → *(arr+i)` and `i[arr] → *(i+arr)`

▼ Difference between pointers and array

- Size of both arr is base address and size of pointer is 8
- arr = arr+1 is not possible but p =p+1 is possible

▼ Char arr

i	s	h	a	n

```
char ch[10] = ishan;
char* c = ch;
cout << c; // output -> "ishan". Zero position se le kr puri
cout << *c; // output -> i
cout << c+2; // output -> "han". second position se le kr pur
```

```
char ch = 'k';
char* cptr = &ch;
cout << cptr; //output -> k^^^. i.e k and garbage value till
```

▼ Difference between int array and char array

- Hence cout ka implementation is different for int and char array, int me base address print krega par char me puri string corresponding to that address print krta h
- &ch will print base address and ch will print full string

Pointer with function

- When you pass array in function so its pointer is passed, so if I change anything in array so it will be change in the main array due to pointer that's why it is known as pass by reference.