Pointers

```
int *ptr = &x;   → address of operator
   ↳ Dereference operator
int *ptr; // Bad Practice.
```

→ Null pointer

```
int *Ptr = 0; // Here, Ptr is a null pointer
```

→ copying pointers

```
int num = 10;
int *p = &num;
int *q = p; // Here, we have copy p in q
            so, q will behave like p.
```

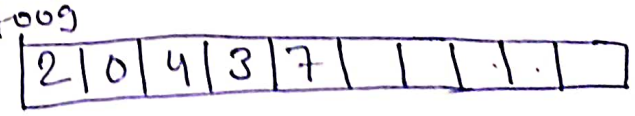
→ pointer arithmetic

```
int num = 10;
int *ptr = &num;
(*ptr)++; // It will increase 10 to 11
ptr++; // It will increase the address.
```

→ ARRAY

```
int arr[0] = {2,0,4,3,7};
```

In memo, 7009



arr

in symbol table,

arr | 7009 |

arr → 7009 → mapped

→ we can't change it.

```
cout<< arr; // print address of 1st element
cout<< &arr; // print address of 1st element
cout<< &arr[0]; // print address of 1st element
cout<< *arr; // value at address of 1st element
cout<< *arr +1; // print 3
cout<< *(arr+2); // print 4
cout<< arr[2]; // print 4
```

Formula,

$arr[i] = *(arr+i)$ or $i[arr] = *(i+arr)$

```
cout<< 3[arr]; // print 3
```

→ pointers array

```
int arr[5] = {1,3,5,9,0};
int *ptr = &arr[0];
cout<< *ptr; // print 1
cout<< *arr; // print 1
cout<< *(ptr+1); // print 2
cout<< *(arr+1); // print 2
```

→ Array is not assignable

```
int arr[3] = {1,2,4};
arr = arr+1; //It will show error because
                arr is not assignable.

int *ptr = &arr[0];
ptr = ptr+2;
cout << *ptr; // print 4
```

→ character array

```
int arr[5] = {1,2,3,4,5};
char ch[6] = "abcde";
cout << arr; // will print address of first
                element of array.
cout << ch; // will print whole string (abcde)
               because character array is
               implemented diffrently.

int *ptr = &arr[0];
char *cptr = &ch[0];
cout << ptr; // print address of 1st element of array
cout << c; // print abcde (start from 1st address
                          and will stop at end of
                          string ('\0'))
char temp;
char *p = &temp;
cout << *p; // print t include some extra character
               because null character is not
                        defined here.
```

→ Pointer array in function

```
int sum(int *arr/arr[], int n){
    sum = sum + i[arr];
}
```

// *arr and arr[] both will take the address of 1st element of array.

```
int main(){
    int arr[10] = { 2,5 6,1,3,8,9};
    sum(arr,6); // print sum upto index 5th
                   from index 0
    sum(arr+2,6); // print sum upto index 5th
                     from index 2.
}
```

→ Double Pointers

```
int n = 10;
int *ptr = &n;
int **pptr = &ptr; // created double pointers.
```