

Chapter 10

- Pointer : variable hold Address .

We should declare pointer of same data that we want to carry it's Address ----> this make compiler can allocate places in memory.

Ex.

```
int var, *ptr;
```


```
Ptr = &var;
```

Note: (*) prefer to be beside datatype

Ex.

```
Int *ptr, *a,*b,...etc (way to declare many pointers).
```

```
Cout << *ptr ; value of place pointer point to it
```



- Point should point to Address has value of same datatype of pointer

Void *ptr ----> compile will detect datatype of pointer

(point to any datatype)

- Reinterpret cast<datatype *>(datatype) (casting)

من الى

Since : name of Array consider pointer to first element

So: name + i ----> point to element i in array

So: *(name + i)----> gives value of this element

➤ Pointer constant and variable

Int a[] *(a++) ---->(forbidden) Bec. (a) is name of array which point to array which is constant (can't change const)

➤ Function and pointers

Fun(double*) ----> you should passing Address to get by pointer

Main:

Double x;

Fun(&x) ----> as pointer carry address

Func(int&var)	Func(int var)	Func(int * var)
Take Address from var and give it to (pointer)	Take copy from value of var	Take Address as var is pointer and give it to function
Change value of var	No change of var	Change value of var place

➤ Passing Array

Func(int a[])


Main---> func(a);

Use pointer ---> pass pointer has Address of Array and can use it or change it .

Ex.

Func(int *ptr)

Main----> func(a) (array)



----> Note : char str [] char* str

 Const pointer  variable pointer

➤ Const Modifier and pointer

- const int* ptr (const value) قيمة المكان نفسه ثابتة
- int* const ptr the pointer is const (can't change) but value of Place can change

To understand : Read from Right to Left .

➤ Array of pointer

- Memory management new , delete

New ----> obtain memory from operating system and Return pointer to it .

Ex.

```
Char* ptr;
```

```
ptr = new char [ x ];
```

This way sit char array in memory with x size and give it's address to ptr

Delete ----> return memory to operating system

-operating system get memory after terminat.

But if function terminate the pointer (declare in function) will end but memory not back the memory to operating system

So: we should use delete operator

delete [] ptr ----> to delete Array

delete ptr ----> to delete pointer point to value

Note : classes should delete All pointer used in it before terminate the class (make that in Destructor)

➤ Pointer to object:

Referring to members of object : use (membership access operator) (->)

Ex. Class dist { };

Main: dist* ptr;

ptr = new dist;

ptr -> func_of_dist();

by this way we use function of dist class.

`dist & obj = *(new dist)` ---> (new) take places in memory equal to size of dist and gives address to obj so we made obj by reference to can git address. (rarely use)

➤ Array of pointers to obj

Class A

Main--> `A* ptrarr [100];`

`Ptrarr[5] = new A;`

`Ptrarr[5] ->funcA();`

In class---> can make pointer to an object of same class but can't make obj of same class

Ex.

Class A {

`A* ptr; (True)`

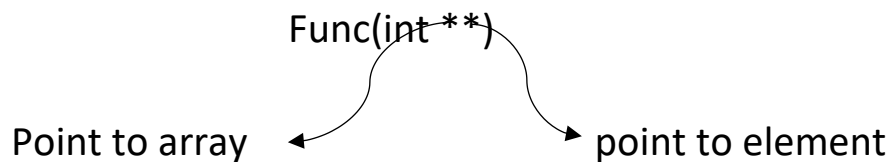
`A obj ; (False)`

}

➤ Point to pointers

Ex.

`Int* pt[];`



➤ Deleting Array of pointers

```
Int* a[ ];
```

```
For(int i=0 ; i < Array size ; i++ )
```

```
    Delete a[ i ];
```

```
Delete [ ] a;
```