# Pointers in Cpp :-

→ Pointers are variables which store memory address of another variable.

2 types of variables

(i) Data variables → int x = 10

(ii) Address variables → int * p;

x [10] → 4 byte
200| 201| 202| 203

p [ ] → 4 byte
300| 301| 302| 303

→ ~~How~~ to differenciate data variable and address variable [ Address variables or pointer are prefix with star]

Declaration of pointer

Now

p = &x; — Initialization of pointer

p [200]
300 | 301 | 302 | 303

```
cout << x;   → 10
cout << &x;  → 200
cout << p;   → 200
cout << &p;  → 300
cout << *p;  → 10  → Dereferencing of pointer
```

What is the purpose of pointer, use of pointer.

```
main()
{
  ≡
}
```

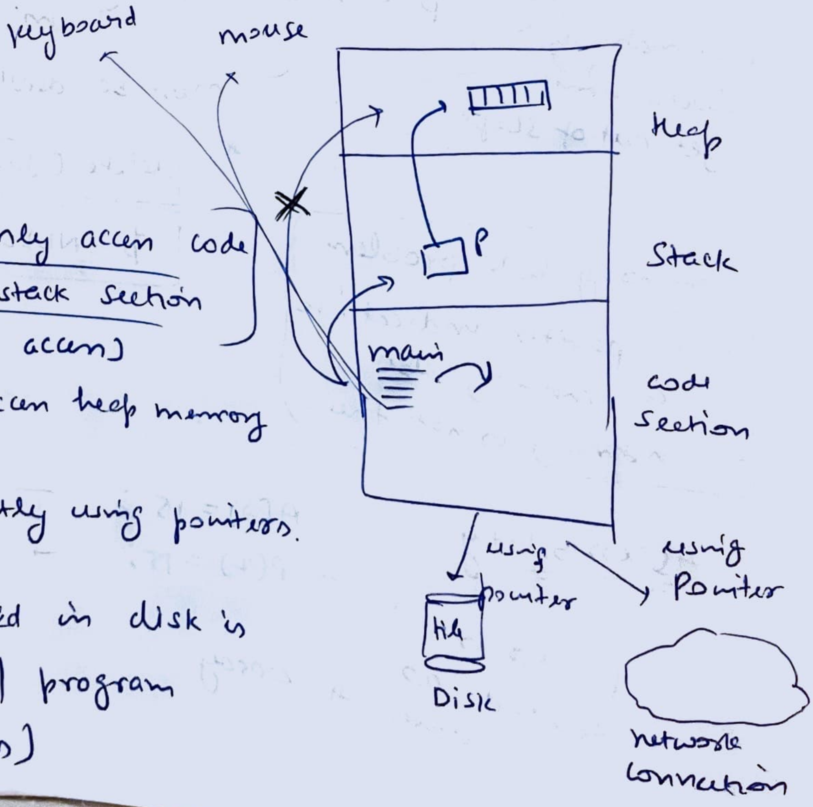
* [main function can only access code section memory and stack section memory [direct access]

× can program directly access heap memory

→ No

But can access indirectly using pointers.

→ How can this file stored in disk is accessed by main fun| program [with help of pointers]



→ if there is a network connection, a mouse, a keyboard, or a monitor, ~~a~~ printer all these things are accessed by program inderictly with help of pointer.

eg cin, cout

→ In c-sharp, Java → no pointer, so they don't allow you to access these devices through your program

can access device using [JVM, commam languangue run-time → c-sharp dot net]

## Heap Memory access / allocation.

→ memory is allocated dynamically,

→ size of memory required in heap is decide at run time not comple time

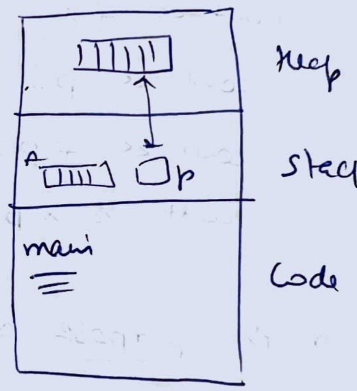
```
main()(
    int A[5] = [1, 2, 3, 4, 5)
        ↳ created in stack

    [ int * p;
        p = new int[5];
                    ↳ in heap
            ↳ must be deallocate / delete
      × delete []p;
        p = NULL;
```

automatically delete when get out of slope

$\boxed{\text{delete []p;}}$

$\boxed{\text{p = NULL;}}$

(If we write this then above memory will be still there)



memory leak problem
→ no pointer indicate to a memory also
→ memory is not free.

Accessibility

A[2] = 15; → direct access

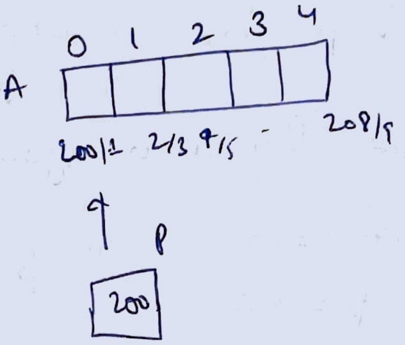
P[2] = 15; → can access

← Pointer is treated as name of array

## Pointer arithmetic

- what are the arithmetic operations allowed on pointer.

```
int A[5] = [2, 4, 6, 8, 10]
int * p = A;
```



→ There are 5 operations allowed on pointers.

(i) p++; → pointer move to next location.

(C → C++) next language

→ it is integer pointer so it will increment by 4 byte.

→ if it is char pointer then it will increment by 1 byte when we do p++.

(ii) p--; → pointer will move backward.

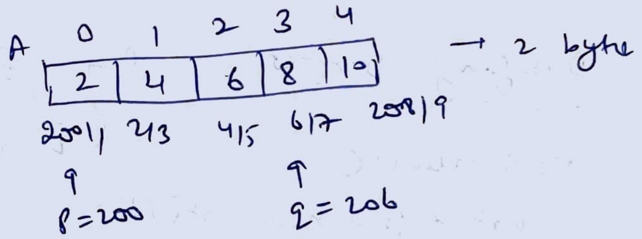
(iii) p = p+2; → move 2 integer place

sy p = 200|2 then p ⇒ 2004|5

(iv) p = p-2;

(v) int d = q-p;

$\boxed{3}$, $\frac{(206|7 - 201|3)}{2}$



→ 2 byte

$= \frac{(\text{add1} - \text{add2})}{(\text{data type of pointer})\text{- size}}$

* $\boxed{\text{int * q = \& A[3];}}$

## Problems using pointers

→ if pointer may not use correctly, program may crush due to run-time error.

→ get a error of run-time at user end, is like a user purchase car, but complain about it's features.

(i). un initialized ptr

(ii) Memory leak } careless run of a program.

(iii) Dassgling pointer

int * p; → Declaration of pointer

* p = 25; → want to store value in pointer

But where it is referencing ?? at some garbage address

(i) int x = 10;
p = &x;

(ii) p = (int *) 0x5638;
hexadecimal

(iii) p = new int [5];

(ii) **Memory leak** ( heap memory)

```
int * p = new int [5];
   } used it
   // now not need it
 p = NULL;
```

delete [] p;

p ✗→ | | | | |

p=0;  p=nullptr;
**

(iii) **Dangling pointer**

P → |7|2|~~|~~| | |
q

```
void main() {
   int * p = new int[5];
      :
   fun (P);
   cout << *p;
}
```

```
void fun ( int * q)
{ ...

  delete []q;
}
```

→ now pointer p is pointing on a location which is no more belonging to program.

**Reference** : Powerful feature of C++, not in any other language

```
main() {
   int x = 10;
   int & y = x;
```

x / y
| 10 |
200101

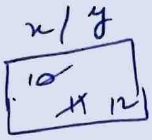
[* nickname of x]

→ if we an variable as reference, we must initialize at same time only.

Example

(i)

```
main(){
  int x = 10;
  int &y = x;
  x++;
  y++;
  cout << x << " " << y;
}
```



12 12

(ii)

```
        int x = 10;
        int &y = x;
        int a;
R-value ←  a = x;
l-value    x = 25;
```

→ (l-value of x)

→ y is not occupying any memory

Same car

R-value → constant value → data of x

l-value → variable → address of x.

* &y = a; ✗

[once you gave a name to variable as reference, we can't assign to any other]