Initialisation List

basics — `int a = 5;` ⟶ `int a;` `a = 5;` | 'a' is created with garbage value then we changed garbage value with '5'.

in Case of Constant → `int Const b = a;` ⟶ `int Const b;` ⊗ `b = a;` ⊗

Creating 'b' with a garbage value is not allowed, as it is meaningless to do so.

in Case of References → `int i = 5;` `int &j = i;` → 5 (i, j)

`int &j;` ⊗ `j = i;` ⊗

⟶

```
Class Student {
    Public:
        int age;
        Const int RollNumber;
};
int main() {
    Student S1;
    S1.age = 20;
    S1.RollNumber = 101;
```

dry Run — default Constructor

S_1: age = //, Roll.No = // ← `Student S1;`

// → garbage

S_1: age = 20, Roll.No = // ← `S1.age = 20;`

&

⟶ Must give "Error".

⟶ but, on Execution we will get Error at "Student S1;" before only, Cause we are not allowed to Create Const with "garbage" value

→ Now if we think of doing, Const int RollNumber = 101; , it is useless as all Students will have same RollNumber then.

→ one feasible Sol^n is, Call the Constructor & in Constructor we need to chage it, as Constructor is must while object is created.

So, Lets try this —

```
Public:
  int age;
  Const int RollNumber;

  Student(int r) {
    RollNumber = r;
  }
```

doing this, we get Error —
" Constructor for 'student' must explicitly initialize the Const member RollNumber"

Lets understand the Error ⟶

```
int a;
a = 5;
```
⟶ we dont write as
```
int a;
int a = 5;
```
because, 'int a;' Leads to creation of 'a' in memory we need not to create again.

in our Code ⟶

```
Const int RollNumber;
&  RollNumber = r;
```
⟶ this shows, that RollNumber is created in memory before only with some garbage value, & we are trying to chage that garbage with "r".

⟶ & to Solve now this issue we need to use, Initialisation list ⟹

=> what initialisation list will do?

→ initialisation list, Jo bhi Constant data members hai Just like "Roll Number" unko Constructor ke andar Jane se Pahale hi (i.e, at the time of memory allocation only) unki value Initialise kar Pae.

→ Lets see how to use it →

```
Public:
    int age;
    Const int RollNumber;

    Student(int r) : RollNumber(r) {

    }
int main() {
    Student s1(101);
    s1.age=20;
}
```

Const int RollNumber = r;

& this works fine

In short NOTE→ whenever we have Constant data members, it is must for us to create a Constructor & use initialisation list.

→ we Can do this as well,

```
Student () : RollNumber(101) {
```

but this is again useless as, Const int RollNumber = 101

→ we can also initialise non-constant datamembers in initisation list as well

Student(int r, int a) : RollNumber(r), age(a) {

& can also write as —

Student(int r, int age) ; RollNumber(r), age(age){

Need not to use this, Infact on using this we might get Error,

age(age)
↙ this ka age
↘ argument wala age

→ & we also do this same in case of Reference Variables.

Const int RollNumber;
int &x; // age Reference Variable

Student(int r, int age) : rollNumber(r), age(age), x(age) {

Now, there is a issue,

what we want ——→ int &x = age;

but what hoppen is ——→ (argument)

i.e, we need to use this here,

⇒ Student(int r, int age) : RollNumber(r), age(age), x(this → age) {

Now it is fine