# Constructors & Destructors in Inheritance

(UNIT-3)

Short forms:

| | | | |
|---|---|---|---|
| Const | → contructor | d1 | → derived 1 |
| dest | → destructor | d2 | → derived 2. |
| inh | → inheritance | der | → derived |
| C/D | → Const & dest. | B | → base. |

---

## 1) Understanding non parameterized Const

- C/D behave differently when inh is involved

```
Base    B() { "Base Const called"; }
  |     ~B() { "B    dest      "    "; }
  |
  ↓
D1      D1 { " d1 cont ....." }
  |     ~d1 { "  .. dest.  ...  }
  |
  ↓
D2      d2 { " d2 Cont    ---- " }
        ~d2 { - - - - - - - - .
```

- Here, each class has a C/D wh. prints "Const called", "dest called";

• Now when obj of d2 is created then • C/D of only d2 is called•

• But the o/Put is

"B const called" } const called top to bottom
"d1 " " "
d2 " "

d2 dest " } dest called bottom to top
d1 " "
B " "

Q) what happens when a derived class obj is created in inh & C/D are involved?
How does C/D beahave in inh?

A) C/D of base class r also called when a derived class obj is created.

Q) In what order $\overset{base}{B}$ & Derived class C/D called?

A) C r called top to bottom i.e. base const first, then derived
Dest r " bottom to top.

Q) Which is the base class of d2
& " " " . . . . . . . d1?

$$\begin{tikzcd}\text{Base} \arrow[d] \\ d1 \arrow[d] \\ d2\end{tikzcd}$$

A) d2 has *only d1* as base class.
B will not be considered as base class of d2.
• d1 has only base as a base class.

Q) When d2 obj is created, then only its base class const should be called i.e only d1 const shud be called. Then why is the const of base called even though it is not base class of d2?

A). Bcoz each const calls its base const autoy.
• So d2 calls d1 const only.
& then d1 const calls base const.
in this sequence.
• So all const r called

Note d2 doesn't call base directly
d2 calls d1, who calls base.

Q) Does const of a derived class called when base obj is created?

A) No when "base obj" is created, then only b const is called. derived const not called.

Q) Why is the const. of base called when der obj is created?

A)

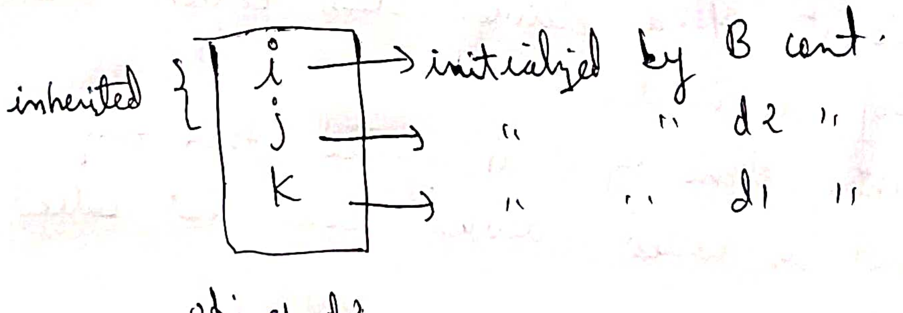
$$\begin{tikzcd}
\boxed{B}\ i \arrow[d] \\
\boxed{d1}\ j \arrow[d] \\
\boxed{d2}\ K
\end{tikzcd}$$

// i, j r der.

Suppose B has a var i
d1 " " " j
d2 " " " K.

- Now in d2 class, there r 3 var
  K is declared in d2 & i, j r inherited.
- So each var shud be initialized by its own const.

∴ Const of all base classes r called when d2 is created to initialize the values of var
o/wise obj will not be initialized completely.



obj of d2

## 2- Using parameterized Const in inh

<u>Rule</u>: When there is a param^3d^ const in B class then der class const must call
* <u>B class const explicitly</u>* & pass values to the B class const.

- der const will call B const using this syntax

```
der( int x, int y) : base (x)  //Pass x to base const.
{
}
```

der const

base const. called explicitly

<u>Eg</u> - In multi-level inh., a chain of const is called.

So d2 will take 3 param, it will pass 2 param to d1

then d1 " " 2 " , it " " 1 " " B

B " " 1 " , it will initialized its own var.

$$\begin{tikzcd}
B\ (x) \arrow[d] \\
d1\ (x, y) \arrow[d] \arrow[u, bend right] \\
d2\ (x, y, 3) \arrow[u, bend right]
\end{tikzcd}$$

Q) WAP to show param^3d^ const in inh?
or
WAP to show how how a derived cont is passing parameters to B class const?

A class base

```
{
        protected:
        int i;
        Public:
        base (int x)
        {
              i = x;
              ~~B~~ cout<<"B const called";
        }
};
Class d1 : public base.
{
        protect:                //x is passed to b const
        int j;                  //do not write int x
        public:                              here
        d1 (int x, int y) : base (x) //No ; here
        {                      //base const explicitly
                                             called here.
              j = y
              cout<<"d1 cnt called";
        //d1 has 2 param
        //y is used to initialize j
        //x is passed to base
        //d1 is explicitly calling B const

        }
};
```

```
class d2 : public d1
{
        Protected:
        int k;                                  Net int x, int y
        public:                                        ↑
        d2(int x, int y, int z) : d1 (x, y)
                             //d1 const is explictly called.
    }
         K = z;
         cout << "d2 cot called";
       //d2 has 3 param
       //z is used to initialze k;
       // x, y r passed to d1 const
      //d2 is explicitly calling d1 const.

    }
    Void get ()
    {
         cout << i << j << k;

    }

}

int main()                   assign to i
 {                                  to j
         d2 obj_d2 (10, 20, 30);        → K
         obj_d2.get();

 }
```

Explanation

```
10 is assign to i   // b (10)          i=10
                        ↑     ↑
10 is passed to
       b const.     // d1 (10, 20)
20 is assigned to j                     j=20
                        ↑      ↑
10, 20 r passed to
        d1 const    // d2 ((10, 20), 30)
30 is assigned to K                      K=30
```