Copy Constructor

Q What is a CC? Why is it used? What is default CC?

A • Every class has a default CC.
• It is provided by the compiler
• It is used to copy one obj to another.

For eg: student obj1;
obj1.set(1, " ", " ");
Student obj2 = obj1;
//Create obj2 & copy obj1 to obj2
//obj1 is copied to obj2 using the def. CC

Q How does you call the CC?
or
In what conditions is the CC called?

A A CC is called in these 3 cases:
1) when initializing one obj using another obj

Eg Student obj2 = obj1;

• Here obj2 is created & initialized using obj1

So, Obj1 is copied to obj2 using def. CC

• Def CC is called implicitly here to copy obj1 to obj2.

• Note*_ Copy is not done by = operator, rather copying is done by CC.

2) Calling the CC explicitly.

```
student obj2(obj1);
```

• The CC is called explicitly like this.

• CC is called & copies obj1 to obj2

3) Passing obj in fun

```
void student :: add_obj (student p01,
                         student p02)     // fun Def
{
    ≡
}

obj3. add_obj (obj1, obj2)     // fun call
```

Copy obj1, obj2 to p01, p02 using CC

• When obj r passed as arguments, they r copied using CC.

Eg obj1 & obj2 r copied to P01 & P02 using CC

Q Suppose there is a parameterized const in class. Can we create obj like this

1) student obj1;
2) studt obj2 = obj1;
3) studt obj2 (obj1);

A 1) No, obj can't be created like this. Values must be passed while creating obj1 like this obj1(1, 11, 11); Here parameterized const will be called.

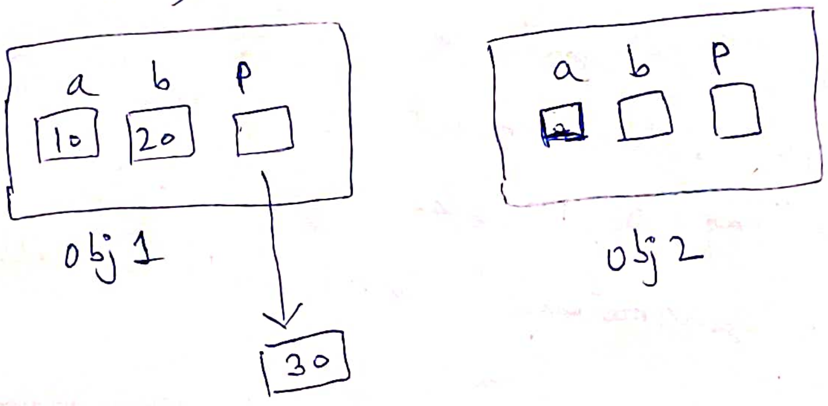
2) Yes, obj2 can be created this way. No need to pass values when creating obj2, because parameterized const will not be called, rather CC will be called.
Parameterized const will be ignored.

3) Yes. Similarly, here CC ~~ta~~ is called explicitly.
So parameterized Const. will be ignored here.

Q what problem may arise when using def CC?

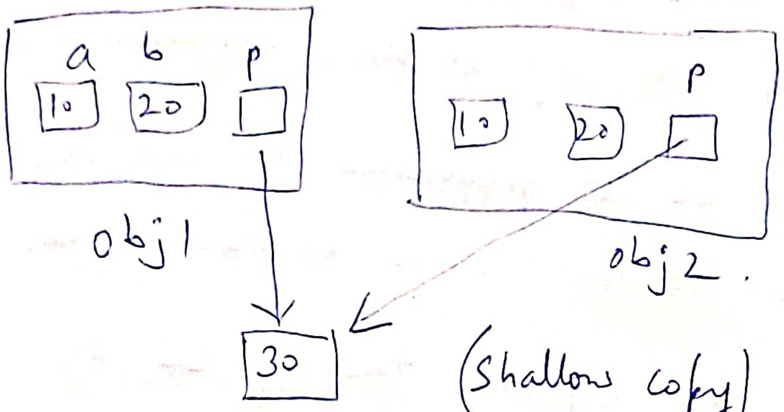
A · A def CC produces a shallow copy of 1 obj to another.

· This may cause problem when copying ~~1 obj to another~~ pointers



· Suppose obj1 is copied to obj2 using def. CC
P is a pointer & it contains an add.

• During a shallow copy - address in p is copied to another p.
So both will point to same address



(Shallow copy)

• Now, the problem with shallow copy is that, if obj1 frees m/m for p, it will be freed for obj 2 also, wh. is not correct.

conclusion :-
The issue with shallow copy (using def cc) is that both p are pointing to same m/m loc.

To avoid this issue, cc is overloaded & user defined cc is created.

Overloading the Def. CC

Q What does overloading def. CC mean ?

A • ODCC means to replace the def CC by a user defined CC

• Def CC is provided by compiler User defined CC is created by user

• If we replace the def CC by a user defined CC, then it is called ODCC.

• ODCC simply means creating a user defined CC.

Q Why is the def CC overloaded?

Or

Why do we need to create a user defined CC?

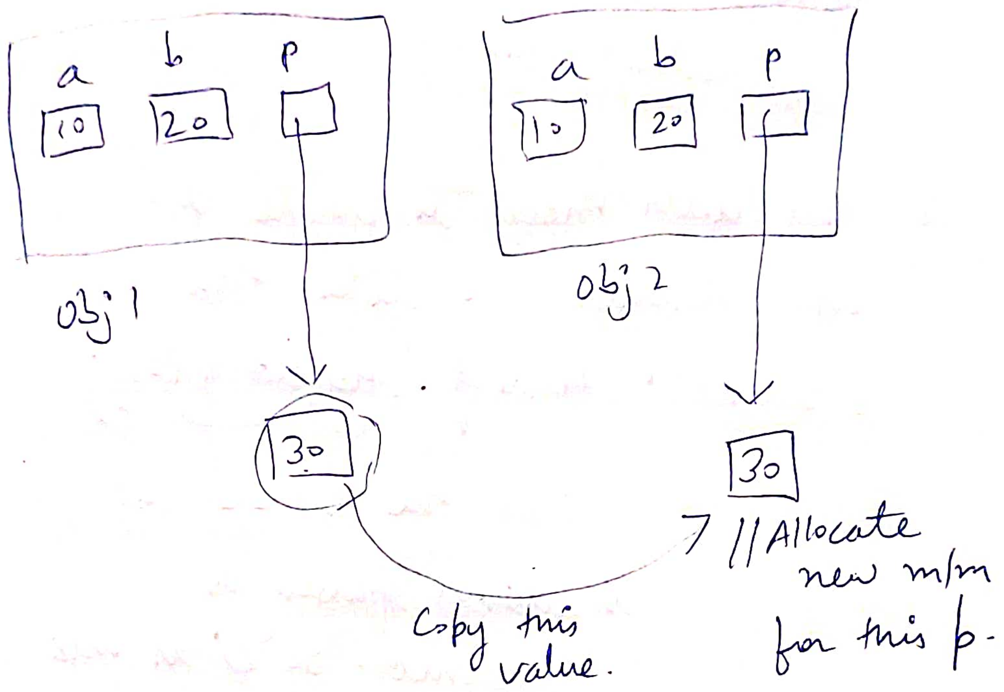
A. The def CC makes a shallow copy

• This causes issue with pointer

• To avoid this, user defined CC is created.

- The user defined CC makes a deep copy of obj, so issue with pointer doesn't arise.

- Deep copy using user defined CC



Deep copy means :-

1) User defined CC will allocate new m/m for p in obj 2

2) Then, it will copy the value 30 to this new m/m

3) Now, both p do not point to same m/m loc

So, if obj 1 frees m/m for P, it will not affect obj 2's p.

Q How is overloading of ~~Def~~ Def CC done?

A 1) You will need to create a blank const. for creating obj like this

student obj1;

2) You will need to create a user defined CC like this

$$\text{Student :: Student}\left(\underbrace{\text{student}}\ \&\ \text{P01}\right);$$

// In the UDCC, obj is always passed as reference. So "&" is used by ~~obj~~ P01.

3) Suppose fun call is like this :-

$$\begin{tikzcd}
\text{student obj2} \arrow[d] & \text{= obj1;} \arrow[d] \\
\text{obj2 is calling obj} & \text{obj1 will be passing obj} \arrow[d] \\
& \text{This will be taken by a reference in fun param.}
\end{tikzcd}$$

- Now obj1 needs to be copied to obj2
- Var of obj2 r rn, m1, m2;
  " " obj1 r po1.rn, po1.m1, po1.m2;

So do this,

```
rn = po1.rn;
m1 = po1.m1;
m2 = po1.m2;
```

- ∴ Fun def for overloaded CC will be

```
student :: student(student & po1)
{       rn = po1.rn ;
        m1 = po1.m1 ;
        m2 = po1.m2;

}
```

Q The complete code.
Overload the def. CC for student class.

A
```
#...
using...
class student
{
    int rn, m1, m2;
    Public:
    void set(int x, int y, int z);
    void get();
    ~~void~~
    // There is no parameterized
     const.
   // You need to create 2 const
   // 1- Blank const (Non Parametized)
   // 2- User defined CC

   // Declare a non parameterized
   // blank const.

   student();

   // Declare the UDCC

   Student(studt & P01);

};
```

```
Void student :: set (int x, int y, int z)
{
        rn = x;
        m1 = y;
        m2 = z;
}

void stud :: add ( )
{
        cout << m1 + m2;
}

student :: student ( )
{
        // Define the blank const
}

// Define the user defined CC.
student :: Student ( Student & P01)
{
        // Copy details of ~~copy ob~~ ~~calling obj~~
        // ~~to~~ Passing obj to calling obj
        rn = P01.rn;
        m1 = P01.m1;
        m2 = P01.m2;
};

int main ( )
{
        // Create obj1. The non parameterized
        // Const will be called for this obj.
        student obj1;
        // Assign values
            obj1.set (1, 11, 11);
//copy obj1 to obj2 using the
//user defined cc
student obj2= obj1;

 obj2.get ();
 obj2.add ();

}
```

<u>o/p</u> :-

```
1,11,11
22.
```