Constructors

15 July 2024
11:06

Default Constructor :-

Default constructor is the constructor which doesn't take any argument. It has no parameters .

if we do not define any constructor explicitly, the compiler will automatically provide a default constructor implicitly. The default value of variables is 0 in case of automatic initialization

Parameterized Constructors :-

It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created. To create a parameterized constructor, simply add parameters to it the way you would to any other function. When you define the constructor's body, use the parameters to initialize the object.

When an object is declared in a parameterized constructor, the initial values have to be passed as arguments to the constructor function. The normal way of object declaration may not work. The constructors can be called explicitly or implicitly.

Example e = Example(0, 50); // Explicit call
Example e(0, 50); // Implicit call

Copy Constructor :-

A copy constructor is a member function which initializes an object using another object of the same class.

It is necessary to pass object as reference and not by value because if you pass it by value its copy is constructed using the copy constructor. This means the copy constructor would call itself to make copy. This process will go on until the compiler runs out of memory.

Note :-

When the **parameterized** constructor is defined and no default constructor is defined explicitly, the compiler will not implicitly create the default constructor and hence create a simple object as: Student s; will flash an error.

Shallow Copy Vs Deep Copy :-

To understand this first we have to consider different types of copy constructor :-
Default copy constructor(compiler generated) , user defined copy constructor

Default copy constructor

```
#include<iostream>
using namespace std;
class A
{
    public :
};
int main()
{
    A obj1;
    A obj2(obj1); //object2 copy object 1 (invoke default constructor)
    return 0;
}
```

user defined copy constructor

Constructor defined by user is called user defined copy constructor.(if we add our own definition of copy constructor)

```
#include<iostream>
using namespace std;
class A
{
    public :
    A()
    {

    }
    A(A &obj){
        cout<<"copy constructor"<<endl;
    }
};
int main()
{
    A obj1;
    A obj2(obj1);
    return 0;
}
```

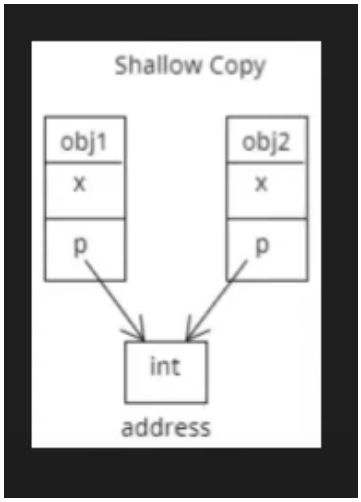
QUESTION:- Why do we need user defined copy constructor if compiler itself generate copy constructor?

To explain this we will create two member variable(data members) -(includes normal variable as well as pointer variables) -Initialise both the variables in default constructor and print

1. Check behaviour using compiler generated copy constructor.

   In case of compiler generated copy constructor

```
 compiler generated copy constructor
    A(const A & obj)
    {
     x = obj.x;
     p = obj.p;    pointer variable points to the same memory address i.e, both
the pointer
                   in obj 1 and obj 2point it the same memory address which we
created in
                   default constructor
    }
```



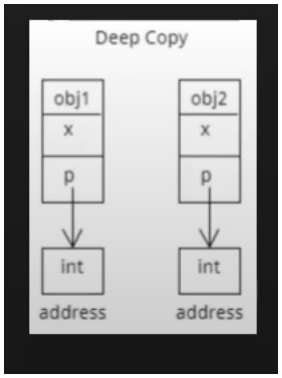
In compiler generated copy constructor shallow copy is performed because of this pointer variables.

Deep Copy:-

For pointer variable we need to create a new memory and then assign the value.

```
//user defined copy constructor
    A(A &obj){
        cout<<"copy constructor"<<endl;
        x= obj.x;
        p = new int ;    //allocate memory for integer pointer p in new object


        *p = *(obj.p);   pointer p in obj 1 and obj 2 pointed to the different
memory
                         locations  (copies the value pointed by obj.p into new
allocated memory)
    }
 //
```



With user defined copy constructor even the value of pointer variable in obj 2 is 5. (Unchanged- **Deep copy**)

Learnings:-

Shallow copy is performed in compiler generated or default constructor.

Deep copy is performed in user defined copy constructor.

QUESTION:- Why do we need user defined copy constructor if compiler itself generate copy constructor?

If a class don't have any pointer member variable , then compiler generated copy cons in enough

If a pointer variable present then we use user defined copy constructor.

Differences between them:-

Shallow Copy	Deep Copy
Shallow Copy stores the references of objects to the original memory address.	Deep copy stores copies of the object's value.
Shallow Copy reflects changes made to the new/copied object in the original object.	Deep copy doesn't reflect changes made to the new/copied object in the original object.

Shallow Copy stores the copy of the original object and points the references to the objects.	Deep copy stores the copy of the original object and recursively copies the objects as well.
A shallow copy is faster.	Deep copy is comparatively slower

That’s why obj 2 also got modified.