📘 OBJECT ORIENTED PROGRAMMING

Topic 3: Constructors & Deep vs Shallow Copy

🔥 ULTRA FAST CONCEPT RECAP

Term	Meaning
Constructor	Special function that runs automatically when object is created
Default Constructor	Constructor with no parameters
Parameterized Constructor	Constructor with arguments
Copy Constructor	Copies one object into another
Shallow Copy	Copies memory address (shared memory)
Deep Copy	Copies actual data into new memory

Now let's understand everything step by step 👇

1.WHAT IS A CONSTRUCTOR?

🔷 Definition

A **constructor** is a special member function of a class that:

- Has the same name as the class
- Runs automatically when object is created
- Initializes data members

🔷 Why We Need Constructor?

Without constructor:

- Data may remain uninitialized
- Object may contain garbage values

Constructor ensures object starts in a valid state.

🔷 Basic Syntax

```
class ClassName
{
public:
    ClassName()
    {
        // initialization code
    }
};
```

2.DEFAULT CONSTRUCTOR

🔷 Definition

A constructor that takes **no parameters**.

If you do not write any constructor, C++ automatically creates one.

🔷 Example 1

```
#include <iostream>
using namespace std;

class Student
{
public:
    string name;
    int age;

    Student()   // Default constructor
    {
        name = "Unknown";
        age = 0;
    }

    void display()
    {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
};

int main()
{
    Student s1;   // constructor runs automatically
    s1.display();

    return 0;
}
```

🔎 Output:

```
Name: Unknown
Age: 0
```

Constructor ran automatically.

3.PARAMETERIZED CONSTRUCTOR

🔷 Definition

A constructor that takes arguments to initialize object with specific values.

🔷 Example

```
#include <iostream>
using namespace std;

class Student
{
public:
    string name;
    int age;

    Student(string n, int a)   // Parameterized constructor
    {
        name = n;
        age = a;
    }

    void display()
    {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
};

int main()
{
    Student s1("Ali", 20);
    Student s2("Sara", 22);

    s1.display();
    s2.display();

    return 0;
}
```

Now each object has its own initialized values.

4.CONSTRUCTOR OVERLOADING

You can create multiple constructors with different parameters.

```
class Example
{
public:
    Example()            // default
    {
        cout << "Default Constructor\n";
    }

    Example(int x)       // parameterized
    {
        cout << "Parameterized Constructor\n";
    }
};
```

5.COPY CONSTRUCTOR

🔷 Definition

A copy constructor creates a new object by copying another object.

It is called when:

```
ClassName obj2 = obj1;
```

🔷 Syntax

```
ClassName(const ClassName &obj)
{
    // copy logic
}
```

🔷 Example

```
#include <iostream>
using namespace std;

class Student
{
public:
    string name;

    Student(string n)
    {
        name = n;
    }

    Student(const Student &s)   // Copy constructor
    {
        name = s.name;
    }
};

int main()
{
    Student s1("Ali");
    Student s2 = s1;   // Copy constructor called

    cout << s2.name;

    return 0;
}
```

6.SHALLOW COPY

🔷 What is Shallow Copy?

It copies pointer values.

Both objects share same memory.

This can cause serious problems.

🔷 Example of Shallow Copy Problem

```
#include <iostream>
using namespace std;

class Example
{
public:
    int *ptr;

    Example(int val)
    {
        ptr = new int;
        *ptr = val;
    }
};

int main()
{
    Example obj1(10);
    Example obj2 = obj1;   // Shallow copy

    *obj2.ptr = 50;

    cout << *obj1.ptr;   // Output?
}
```

⚠️ Problem

Both objects share same memory.

Changing obj2 also changes obj1.

7.DEEP COPY

🔷 What is Deep Copy?

Deep copy:

- Allocates new memory
- Copies actual data
- Objects become independent

🔷 Example of Deep Copy

```
#include <iostream>
using namespace std;

class Example
{
public:
    int *ptr;

    Example(int val)
    {
        ptr = new int;
        *ptr = val;
    }

    // Deep Copy Constructor
    Example(const Example &obj)
    {
        ptr = new int;
        *ptr = *obj.ptr;
    }
};

int main()
{
    Example obj1(10);
    Example obj2 = obj1;    // Deep copy

    *obj2.ptr = 50;

    cout << *obj1.ptr << endl;   // 10
    cout << *obj2.ptr << endl;   // 50

    return 0;
}
```

Now both objects are independent.

🔥 Difference Between Shallow & Deep Copy

Shallow Copy	Deep Copy
Copies address	Copies actual data
Shared memory	Separate memory
Risky	Safe

Shallow Copy	Deep Copy
Default behavior	Requires custom copy constructor

8.IMPORTANT NOTE

If your class uses:

- Dynamic memory (`new`)
- Pointers

Then you MUST implement:

- Copy constructor
- Destructor (next topic)

Otherwise, program may crash.

🧠 BRAINSTORM QUESTIONS

1. Why is constructor name same as class name?
2. When is copy constructor automatically called?
3. What problem occurs in shallow copy?
4. Why is deep copy safer?
5. What happens if we don't write a constructor?

🏁 SUMMARY

- Constructor runs automatically
- Default constructor → No arguments
- Parameterized constructor → With arguments
- Copy constructor → Copies object
- Shallow copy → Shared memory (dangerous)
- Deep copy → Independent memory (safe)