SOS IN COMPUTER SCIENCE & APPLICATION
JIWAJI UNIVERSITY

Class : MBA (E-Commerce) II Semester

Subject : OOPS using C++

Paper Code: (201)

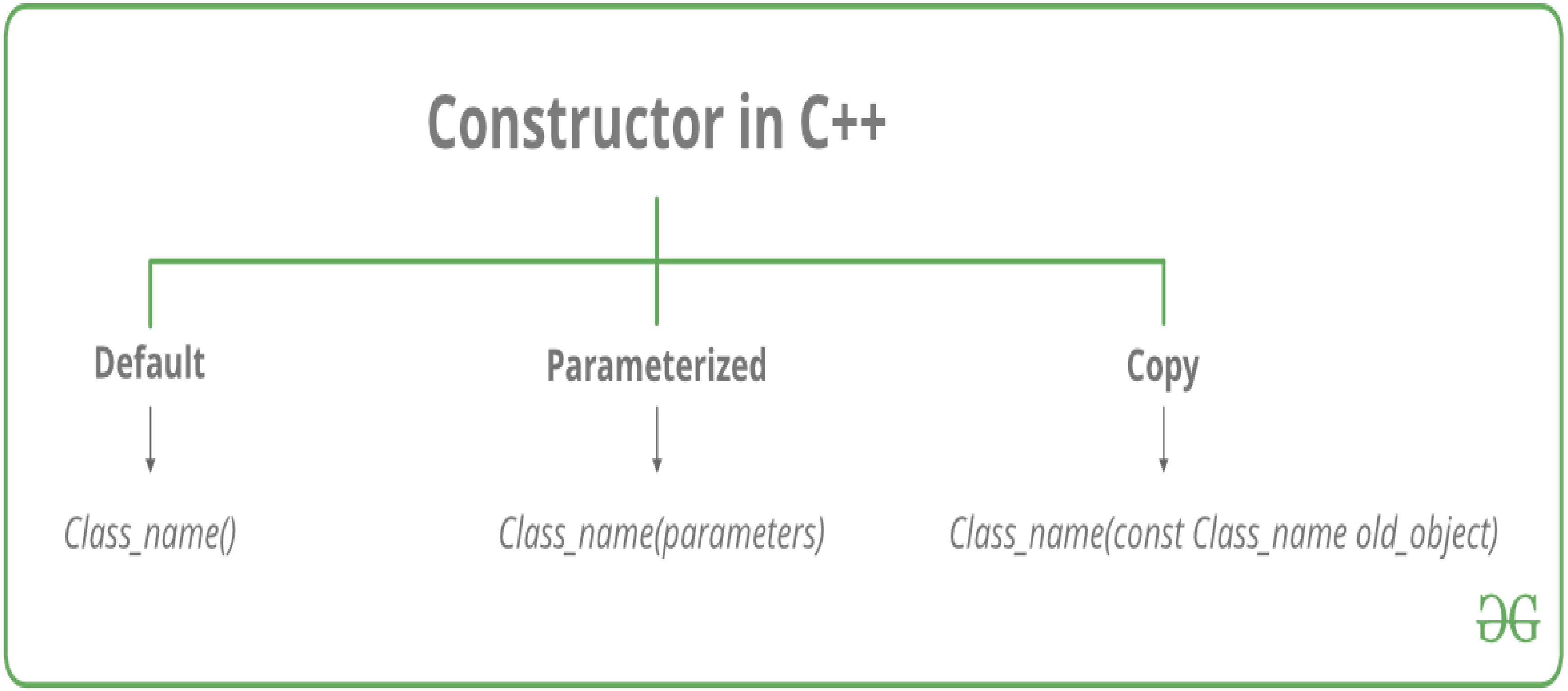
Topic: Constructors in C++

Constructors in C++

- A constructor is a member function of a class which initializes objects of a class. In C++, Constructor is automatically called when object(instance of class) create. It is special member function of the class.
- **How constructors are different from a normal member function?**

- A constructor is different from normal functions in following ways:
- Constructor has same name as the class itself
- Constructors don’t have return type
- A constructor is automatically called when an object is created.
- If we do not specify a constructor, C++ compiler generates a default constructor for us (expects no parameters and has an empty body).

Types Of Constructors



Default Constructor

Default Constructors: Default constructor is the constructor which doesn'ttake any argument. It has no parameters.

```
 // Cpp program to illustrate the
// concept of Constructors
#include <iostream>
using namespace std;

class construct
 {
public:
    int a, b;

    // Default Constructor
    construct()
    {
        a = 10;
        b = 20;
    }
};

```

```
int main()
{
    // Default constructor called automatically
    // when the object is created
    construct c;
    cout << "a: " << c.a << endl
         << "b: " << c.b;
    return 1;
}
```

- Output:

```
a: 10
b: 20
```

- **Note:** Even if we do not define any constructor explicitly, the compiler will automatically provide a default constructor implicitly

Parameterized Constructors

- **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.
- Program is given on **Next Slide** using parametrized Constructor.

```
// CPP program to illustrate
// parameterized constructors
#include <iostream>
using namespace std;

class Point {
private:
    int x, y;

public:
    // Parameterized Constructor
    Point(int x1, int y1)
    {
        x = x1;
        y = y1;
    }

    int getX()
    {
        return x;
    }
    int getY()
    {
        return y;
    }
};
```

Program Cont.

- int main()
- {
-     // Constructor called
-     Point p1(10, 15);
-
-     // Access values assigned by constructor
-     cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();
-
-     return 0;
- }
- Output:
- p1.x = 10, p1.y = 15
- 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**
- **e(0, 50); // Implicit call**

Uses of Parameterized constructor& Copy Constructor

- **Uses of Parameterized constructor:**
  - It is used to initialize the various data elements of different objects with different values when they are created.
- It is used to overload constructors.
- **Can we have more than one constructors in a class?** Yes, It is called Constructor Overloading.
- **Copy Constructor:** A copy constructor is a member function which initializes an object using another object of the same class. Detailed article on Copy Constructor.
- Whenever we define one or more non-default constructors(with parameters) for a class, a default constructor(without parameters) should also be explicitly defined as the compiler will not provide a default constructor in this case. However, it is not necessary but it's considered to be the best practice to always define a default constructor.

Copy Constructor

```
#include <iostream>
using namespace std;
class A
{
   public:
    int x;
    A(int a)                // parameterized constructor.
    {
      x=a;
    }
    A(A &i)               // copy constructor
    {
        x = i.x;
    }
};
int main()
{
  A a1(20);               // Calling the parameterized constructor.
 A a2(a1);                //  Calling the copy constructor.
 cout<<a2.x;
  return 0;
}
```

Output:

```
20
```