Lecture 1: Constructor

Today’s Agenda

Today we are going to cover -

- Constructor
- Types of Constructor
- Default Constructor
- Parameterized Constructor
- Copy Constructor
- Practice Question

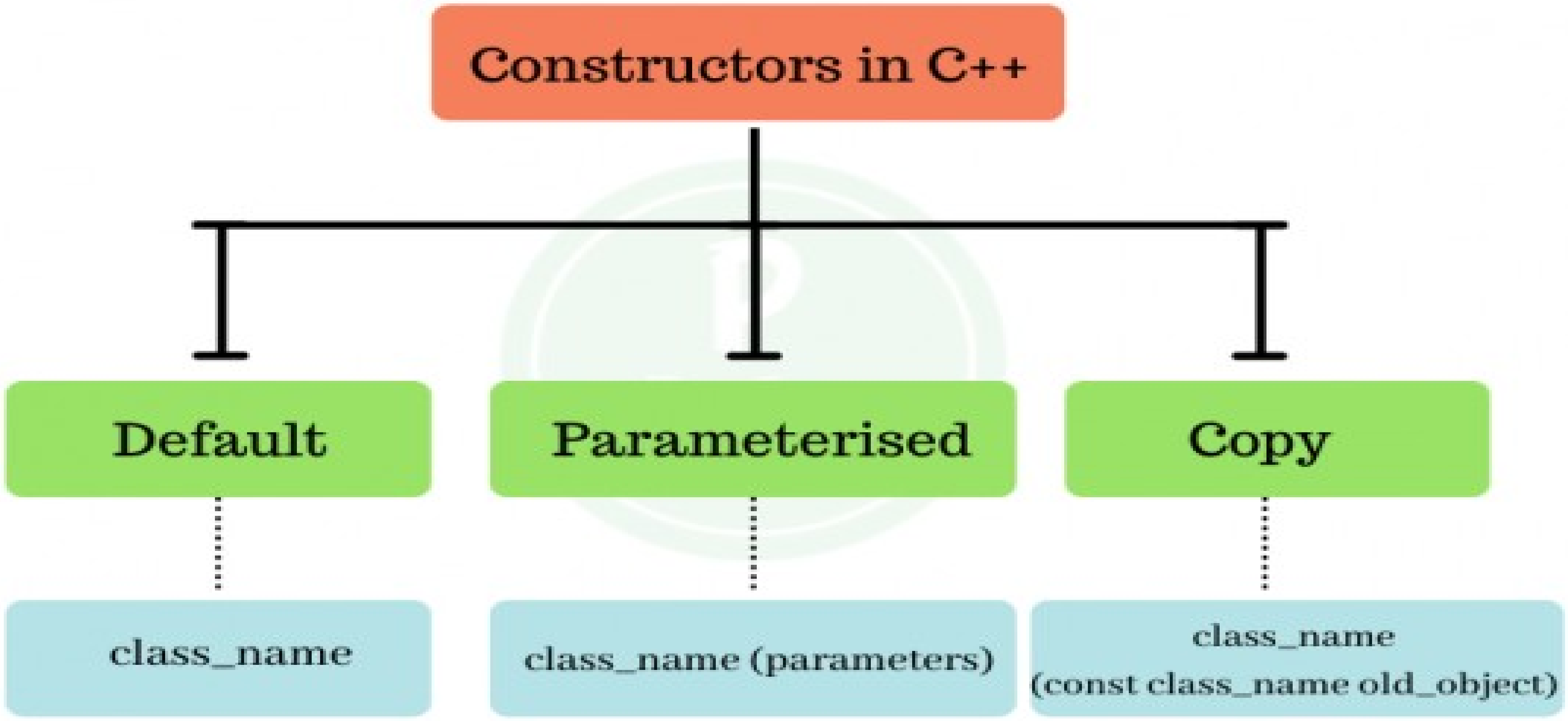
Let’s Get Started-

Constructor

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.

- 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 constructor



Default Constructor

1. Default Constructors: Default constructor is the constructor which doesn’t take any argument. It has no parameters.

```
class construct
{
public:
    int a, b;

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

Default Constructor

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

Default Constructor

```
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
```

Output

```
a: 10

b: 20
```

Even if we do not define any constructor explicitly, the compiler will automatically provide a default constructor implicitly.

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.

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.

Parameterized Constructors

```
class Point
{
private:
    int x, y;

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

Parameterized Constructors

```
int getX()
    {
        return x;
    }
    int getY()
    {
        return y;
    }   };
int main()
{
    // Constructor called
    Point p1(10, 15);
    cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();
    return 0;
}
```

Parameterized Constructors

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

 Example e(0, 50);          // Implicit call
```

We can have more than one constructor in a class it is called constructor overloading.

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.

Copy Constructor

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

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

```
class point
{
private:
double x, y;

public:

// Non-default Constructor &
// default Constructor
point (double px, double py)
{
    x = px, y = py;
}
};
```

Copy Contstructor

```
int main(void)
{

// Define an array of size
// 10 & of type point
// This line will cause error
point a[10];

// Remove above line and program
// will compile without error
point b = point(5, 6);
}
```

Output

```
Error: point (double px, double py): expects 2 arguments, 0 provided
```

Output

```
Error: point (double px, double py): expects 2 arguments, 0 provided
```

MCQ 1

Which of the followings is/are automatically added to every class, if we do not write our own.

(A) Copy Constructor

(B) Assignment Operator

(C) A constructor without any parameter

(D) All of the above

MCQ 1

Which of the followings is/are automatically added to every class, if we do not write our own.

(A) Copy Constructor

(B) Assignment Operator

(C) A constructor without any parameter

(D) All of the above

MCQ 2

```
#include<iostream>
using namespace std;
class Point {
    Point() { cout << "Constructor called"; }
};

int main()
{
Point t1;
return 0;
}
```

(A) Compiler Error
(B) Runtime Error
(C) Constructor called

MCQ 2

```
#include<iostream>
using namespace std;
class Point {
    Point() { cout << "Constructor called"; }
};

int main()
{
Point t1;
return 0;
}
```

(A) Compiler Error

(B) Runtime Error

(C) Constructor called

Guess the output 1

```
class A{
    mutable int a;
public:
    A(){
        cout<<"Default constructor called\n";
    }
    A(const A& a){
        cout<<"Copy Constructor called\n";
    }
};
int main(int argc, char const *argv[])
{
    A obj;
    A a1 = obj;
    A a2(obj);
}
```

Guess the output 1

```
Default constructor called
Copy Constructor called
Copy Constructor called
```

Guess the output 2

```
class A{
    mutable int a;
public:
    A(){
        cout<<"A's default constructor called\n";
    }
    A(const A& a){
        cout<<"A's copy Constructor called\n";
    }
};
class B{
    A obj;
public:
    B(){
        cout<<"B's Constructor called\n";
    }
```

Guess the output 2

```
int main(int argc, char const *argv[])
{
    B b1;
    B b2;
}
```

Guess the output 2

```
A's default constructor called

B's Constructor called

A's default constructor called

B's Constructor called
```

QNA Time

Any Questions ??

Any Questions??

Thank You!

See you guys in next class.