**Class:**

**What?**

Class is a user defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

**When and why?**

When we want many properties and data to be stored in a variable we can use class and the variable is the object in the class.

**How?**

Syntax:

**class** class\_Name {

//access specifier

// Data\_Member

// Member\_Function

};

class\_Name ObjectName;

Accessing data members and member function:

Data member and member function of a class can be accessed using (.)dot operator. Ex, lets say obj is the object name and trying to access the function printData() in the class.

obj.printData();

There are three access specifier,

* public
* protected
* private

**Program**

**#01\_Pgm**

#include<iostream>

#include<string>

class Data

{

public:

std::string name;

void printName(){

std::cout << "Name is " << name << std::endl;

}

};

int main()

{

Data obj;

obj.name = "Kumar";

obj.printName();

return 0;

}

Output:

Name is Kumar

**Constructor:**

Its a special member function that is called by the compiler when ever a object for a class is instantiated.

Types of constructor:

1. Default constructor

2. Parameterized constructor

3. Copy constructor

**why constructor?**

It can be used to initialize the data members of a class.

**Ex:**

class Point

{

public:

Point(){

x = 0.0;

y = 1.1;

}

double x, y;

};

**Default constructor:**

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

**#02\_Pgm**

#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 << endl;

return 1;

}

Output:

a: 10

b: 20

**Parameterized Constructor:**

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.

Uses:

1. It is used to initialize the various data elements of different objects with different values when they are created.   
2. It is used to overload constructors.

**#03\_Pgm**

#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;

}

};

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

**Copy Constructor:**

Its a member function which initialize an object using another object of the same class.

Syntax: ClassName (const ClassName &old\_obj);

#### When is copy constructor called:

1. When object of class is returned by value.

2. When an object of the class is passed (to a function) by value as an argument.

3. When an object is constructed based on another object of the same class.

4. When the compiler generates a temporary object.

#### #04\_Pgm

#include<iostream>

using namespace std;

class Point

{

private:

int x, y;

public:

Point(int x1, int y1) { x = x1; y = y1; }

// Copy constructor

Point(const Point &p2) {x = p2.x; y = p2.y; }

int getX() { return x; }

int getY() { return y; }

};

int main()

{

Point p1(10, 15); // Normal constructor is called here

Point p2 = p1; // Copy constructor is called here

// Let us access values assigned by constructors

cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();

cout << "\np2.x = " << p2.getX() << ", p2.y = " << p2.getY() << endl;

return 0;

}

Output:

p1.x = 10, p1.y = 15

p2.x = 10, p2.y = 15