Constructor

- Constructor is used to ***build the Object***.
- It not only build the object as well as to perform any additional activity at the time of the creation of object such as set the ***Initial state of object's Data Member***, ***Establish the connection with data base***, ***Establish the connection with the file Stream, generating the initial state of GUI*** and any other task whatever the developer wants.
- ***Any activity that you want perform only once while creation of the object.***

e.g.

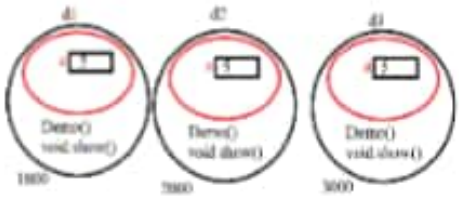
```
#include<iostream>
using namespace std;
class Demo
{
        public:
                Demo()//constructor
                {
                        cout << endl << "constructor execute automatically";
                }
                void show()
                {
                        cout <<endl<< "show";
                }
};
main()
{
        Demo d;
        d.show();
}
```

- In C++ Constructor is a ***special method***.
- It has the ***same name as the class name***.
- Constructor ***does not have return type specification and not even void***.
- If ***constructor*** does ***not execute successfully, the object will not be form***.
- ***Constructors are following type in C++***
  - Default **Constructor** or Zero argument **constructor**.
  - Parameterized **constructor**
  - Copy **constructor**
  - Conversion **constructor**
  - Explicit **constructor**
  - Dynamic constructor

∴ ***Default constructor***

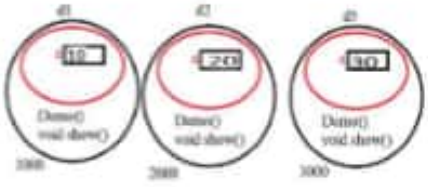
- The default constructor has not argument /parameter.

```
#include<iostream>
using namespace std;
class Demo
{
        private:
                int a ;
        public:
                Demo()
                {
                        a = 5 ;
                }
                void show(){
                        cout <<endl<< " a = "<< a;
                }
};
main()
{
        Demo d1, d2 ,  d3;
        d1.show();
        d2.show();
        d3.show();
}
```



∴ **The parameterized constructor have parameter list.**

```
#include<iostream>
using namespace std;
class Demo
{
        private:
                int a;
        public:
                Demo(int n)
                {
                        a = n ;
                }
                void show(){
                        cout <<endl<< " a = "<< a;
                }
};
main()
{
        Demo d1(10), d2(20) ,  d3(30);
        d1.show();
        d2.show();
        d3.show();
}
```



- If we do not create a constructor explicitly, *then technologies silently place a default constructor.*

```
#include<iostream>
using namespace std;
class Demo
{
        private:
                int a;
        public:
                void show()
                {
                        cout <<endl<< " a = "<< a;
                }
};
main()
{
        Demo d1;
        d1.show();
}
```

- ***We can overload the constructor.***

```
class Demo
{
        private :
            int a;
public:
        Demo()
        {
            a = 5;
        }
        Demo(int n)
        {
            a = n;
        }
        void show(){
            cout << "a = " + a ;
        }
};
main()
{
        Demo d1 , d2(10);
        d1.show();
        d2.show();
}
```

- ***Copy Constructor***
  - ***Copy constructor allow us to create duplicate object.***

```
#include<iostream>
using namespace std;
class Demo
{
        private:
                int a;
        public:
                Demo(int n)
                {
                        a = n;
                }
                Demo(Demo &o)
                {
                        a = o.a;
                }
                void show(){
                        cout <<endl<< " a = "<< a;
                }
};
main()
{
        Demo d1(20);
        Demo d2(d1);
        d1.show();
        d2.show();
}
```

- ***We can define a constructor outside the class.***

```
#include<iostream>
using namespace std;
class Demo
{
    int i;
    public :
    Demo()()
    Demo(int id);
    void show()
    {
            cout << endl << " i = " << i;
    }

};
Demo::Demo(int id)
{
    i = id ;
```

```
}

main()
{
    Demo  d(30) ; // implicit calling of constructor
    d.show();

}
```

- *Constructors can be generated by C++ if they haven't been explicitly defined. They are also invoked on many occasions without explicit calls in your program. Any constructor generated by the compiler will be* **public.**
- *You can't call constructors the way you call a normal function.*
- *They can't be inherited, though a derived class can call the constructors of the base class*