Constructor

A member Function that has the same name as the class name. It is automatically called when object of a class is created in main. The constructor has no return type that means it cannot return any value.

They should be declared in the public section.

They are invoked automatically when the objects are created.

- They do not have return type, not even void.
- They cannot be inherited, though a derived class can call the base class constructor.
- Like other c++ functions, they can have default arguments.
- Constructors cannot be virtual.

Constructors are of 3 types

- **Default Constructor**
- **Parameterized Constructor**
- **Copy Constructor**

1. Default Constructor
   - Default Constructor: A constructor that accepts no parameters is called the **default constructor**.

Syntax

```
Name()
{

}
```

Class Name

Syntax

```
class Name
{
    Name()
    {

        Body of Member function constructor
    }
};
```

Constructor

2. Parameterized Constructor
   - The constructors that take parameters are called parameterized constructors.

Syntax

```
#include<iostream.h>
class item
{
      int m,n;
      public:
      item(int x, int y){
      m=x;
      n=y;
                  }

};
```

Class Name

Parameter

Constructor

Constructor

3. Copy Constructor
   - A copy constructor is used to declare and initialize an object from another object.

Syntax

```
Class_name object_name(parameter)
Class_name object_name=second_obj_name(parameter)
```

Program Default Constructor

```
#include <iostream>
using namespace std;
class Display{            -> Display is name of class
private:
    int a=1;
public:
    Display(){            -> Display() fuction is constructor same name as class name
    cout<<"Value : "<<a;
    }
};
int main()
{
    Display obj;          -> Only one time obj call the Display class and then automatically Display() function execute

}
```

```
"C:\Users\Shahzaib Yaqoob\Desktop\Code_Block\IN\123\main.exe"
Value : 1
Process returned 0 (0x0)   execution time : 0.328 s
Press any key to continue.
```

Code

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

class Display{
private:
  int a=1;
public:

Display(){
  cout<<"Value : "<<a;
  }
};

int main()
{
  Display obj;

}
```

Class Name

Constructor

Class Name and Constructor

Object

Constructor

Constructor overloading

The process of declaring multiple constructor with same name but different parameters. The constructor with same name as class name but it has must differ.

- **Number of parameters**
- **Type of parameters**
- **Sequence of parameters**

Syntax

```
Name (parameter)
{

}
```

Different Parameter or Data type

Syntax

```
class Name
{
    Name ()
    {

        Body of Member function constructor
    }
    Name (parameter)
    {

        Body of Member function constructor
    }

};
```

Program

```
#include <iostream>
using namespace std;
class Book                          ->Book is name of classclass Display
{
private:
int id,page,price;
public:
Book()                          -> Book() fuction is constructor same name as class name
    {
    cout<<"\t\t\t\t\t\t:::::Enter Input:::::"<<endl;
    cout<<"Enter book id : ";
    cin>>id;
    cout<<"Enter book page number : ";
    cin>>page;
    cout<<"Enter book price : ";
    cin>>price;
    }
Book(int ID,int PG,int PR)      ->void is return type Book is constructor with(int ID,int PG,int PR) parameters
                                        and giving data from obj2(2,500,700)
    {
        id=ID;
        page=PG;
        price=PR;
    }
void show()                      -> show() member fuction;
    {
        cout<<"\n\n\n\n\t\t\t\t\t\t:::::Show display:::::"<<endl;
        cout<<"Book ID = "<<id<<endl;
        cout<<"Page number = "<<page<<endl;
        cout<<"Price = "<<price<<endl;
    }
};
```

```
main()
{
Book obj1,obj2(2,500,700);          ->Book obj1 calling first book member function and obj2(2,500,700) calling second member function
obj1.show();
obj2.show();}
```

Constructor

```
"C:\Users\Shahzaib Yaqoob\Desktop\Code_Block\IN\123\main.exe"
                                        :::::Enter Input:::::
Enter book id : 1
Enter book page number : 2
Enter book price : 3



                                        :::::Show display:::::
Book ID = 1
Page number = 2
Price = 3



                                        :::::Show display:::::
Book ID = 2
Page number = 500
Price = 700
```

Code

```
#include <iostream>
using namespace std;
class Book
{
private:
int id,page,price;
public:
    Book()
    {
    cout<<"\t\t\t\t\t:::::Enter Input:::::"<<endl;
    cout<<"Enter book id"<<endl;
    cin>>id;
    cout<<"Enter book page number"<<endl;
    cin>>page;
    cout<<"Enter book price"<<endl;
    cin>>price;
    }
    Book(int ID,int PG,int PR)
    {
    id=ID;
    page=PG;
    price=PR;
    }
    void show()
    {
    cout<<"\n\n\n\t\t\t\t\t:::::Show display:::::"<<endl;
    cout<<"Book ID = "<<id<<endl;
    cout<<"Page number = "<<page<<endl;
    cout<<"Price = "<<price<<endl;
    }

};
main()
{
Book obj1,obj2(2,500,700);;
obj1.show();
obj2.show();
}
```

Constructor

Default Copy Constructor

We initialize the object with another existing object of the same class type for example Book is class name and it have three object obj1 , obj2 and obj3.

Book obj2(obj1) obj2 copy the all data of obj1
Book obj3=obj1 obj3 copy the all data of obj1

Syntax

Class_name object_name(parameter)
Class_name object_name=second_obj_name(parameter)

Program

```
#include <iostream>
using namespace std;
class Book                          ->Book is name of classclass Display
{
private:
int id,page,price;
public:
Book()                          -> Book() fuction is constructor same name as class name
    {
    cout<<"\t\t\t\t\t\t:::::Enter Input:::::"<<endl;
    cout<<"Enter book id : ";
    cin>>id;
    cout<<"Enter book page number : ";
    cin>>page;
    cout<<"Enter book price : ";
    cin>>price;
    }
Book(int ID,int PG,int PR)      ->void is return type Book is constructor with(int ID,int PG,int PR) parameters
                                    and giving data from obj2(2,500,700)
    {
        id=ID;
        page=PG;
        price=PR;
    }
void show()                      -> show() member fuction;
    {
        cout<<"\n\n\n\t\t\t\t\t\t:::::Show display:::::"<<endl;
        cout<<"Book ID = "<<id<<endl;
        cout<<"Page number = "<<page<<endl;
        cout<<"Price = "<<price<<endl;
    }
};
main()
{
Book obj1;                  -> obj1 is object of Book class
obj1.show();
cout<<endl<<endl;
Book obj2(obj1);            -> obj2 copy the all data of obj1
obj2.show();
cout<<endl<<endl;
Book obj3=obj1;             -> obj3 copy the all data of obj1
obj3.show();
}
```

```
"C:\Users\Shahzaib Yaqoob\Desktop\Code_Block\IN\123\main.exe"
                                        :::::Enter Input:::::
Enter book id
1
Enter book page number
2
Enter book price
3



                                        :::::Show display:::::
Book ID = 1
Page number = 2
Price = 3



                                        :::::Show display:::::
Book ID = 1
Page number = 2
Price = 3



                                        :::::Show display:::::
Book ID = 1
Page number = 2
Price = 3
```

Constructor

Code

```
#include <iostream>
using namespace std;
class Book
{
private:
int id,page,price;
public:
    Book()
    {
    cout<<"\t\t\t\t\t:::::Enter Input:::::"<<endl;
    cout<<"Enter book id"<<endl;
    cin>>id;
    cout<<"Enter book page number"<<endl;
    cin>>page;
    cout<<"Enter book price"<<endl;
    cin>>price;
    }
    Book(int ID,int PG,int PR)
    {
        id=ID;
        page=PG;
        price=PR;
    }
    void show()
    {
        cout<<"\n\n\n\t\t\t\t\t:::::Show display:::::"<<endl;
        cout<<"Book ID = "<<id<<endl;
        cout<<"Page number = "<<page<<endl;
        cout<<"Price = "<<price<<endl;
    }

};
 main()
{
Book obj1;
obj1.show();
cout<<endl<<endl;
Book obj2(obj1);
obj2.show();
cout<<endl<<endl;
Book obj3=obj1;
obj3.show();
}
```

Constructor

Destructor

Destructor has no return type and its name is same name as class name but it is automatically executed when an object of the class is destroyed. It cannot accept any parameter.

Syntax

```
~name( )
{

   Body

}
```

Destructor

Syntax

```
class name{

public:
        name( ){

                Body of Constructor

}
        ~name( ){

                Body of Destructor

}
};
```

Class Name

Constructor

Destructor

Constructor

Program

```
#include <iostream>
using namespace std;
class Book
{
private:
int id,page,price;
public:
Book()                          ->constructor
    {
    cout<<"\t\t\t\t\t:::::Enter Input:::::"<<endl;
    cout<<"Enter book id"<<endl;
    cin>>id;
    cout<<"Enter book page number"<<endl;
    cin>>page;
    cout<<"Enter book price"<<endl;
    cin>>price;
    }
~Book()                          -> Destructor
    {
        cout<<"\n\n\n\t\t\t\t\t:::::Show display:::::"<<endl;
        cout<<"Book ID = "<<id<<endl;
        cout<<"Page number = "<<page<<endl;
        cout<<"Price = "<<price<<endl;
    }

};
```

```
main()
{
Book obj1;                  ->Only one time obj1 is calling Constructor Book Member function for input
                              and Destructor ~Book for output
}
```

```
"C:\Users\Shahzaib Yaqoob\Desktop\Code_Block\IN\123\main.exe"
                                        :::::Enter Input:::::
Enter book id
22
Enter book page number
33
Enter book price
44



                                        :::::Show display:::::
Book ID = 22
Page number = 33
Price = 44
```

Constructor

Code

```
#include <iostream>
using namespace std;
class Book
{
private:
int id,page,price;
public:
    Book()
    {
    cout<<"\t\t\t\t\t:::::Enter Input:::::"<<endl;
    cout<<"Enter book id"<<endl;
    cin>>id;
    cout<<"Enter book page number"<<endl;
    cin>>page;
    cout<<"Enter book price"<<endl;
    cin>>price;
    }
~Book()
    {
        cout<<"\n\n\n\t\t\t\t\t:::::Show display:::::"<<endl;
        cout<<"Book ID = "<<id<<endl;
        cout<<"Page number = "<<page<<endl;
        cout<<"Price = "<<price<<endl;
    }

};
 main()
{
Book obj1;
}
```

Constructor

Friend Function

A type of function that has access the private, protected, and public members of a particular class from outside the class. A friend function is used to access all the non-public members of a class. You can use a friend function to bridge two classes by operating objects of two different classes. The private and protected members of any class cannot be accessed from outside the class. In some situations, program may require to access these members. The use of friend's functions allows the user access these members. A function that is declared in class with friend keywords become the friend function of the class.

Program

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

class B        -> class B declare Before the friend function because complier know about
                  class B is declare in Program when freind Function is execute.

class A        -> class A
{
private:
    int a;
public:
    A()         -> Member function
    {
    a=10;
    }
friend void fread_A_B(A,B);     ->  Friend_A_B(A,B) is friend class
                                class A and class B is Freind of Friend_A_B() function and
                                    Share private ,protected and public data in it.
};

class B                             class B
{
private:
    int b;
public:
    B()                         -> Member function
    {
       b=20;
    }
    friend void fread_A_B(A,B);  -> Friend_A_B(A,B) is friend class
                                    class A and class B is Freind of Friend_A_B() function and
                                    Share private ,protected and public data in it.
};


void fread_A_B(A x,B y)         -> fread_A_B() is friend function
                                   A and B is class and  x and y recive obj1 and obj2 data
{
int r;
r=x.a+y.b;                      -> x.a means x recive obj1 a varible which value is 10 and same y.b...

cout<<"sum of a and b from different class : "<<r<<endl;

}

 main()
{
A obj1;
B obj2;
fread_A_B(obj1,obj2);               -> fread_A_B(obj1,obj2) obj1 and obj2 passing obj_data and x, and y recive thier data
}
```

```
"C:\Users\Shahzaib Yaqoob\Desktop\Code_Block\IN\123\main.exe"
sum of a and b from different class : 30

Process returned 0 (0x0)   execution time : 0.409 s
Press any key to continue.
```

Constructor

Code

```
#include <iostream>
using namespace std;
class B;
class A
{
private:
    int a;
public:
    A()
    {
    a=10;
    }
friend void fread_A_B(A,B);
};

class B
{
private:
    int b;
public:
    B()
    {
       b=20;
    }
    friend void fread_A_B(A,B);
};


void fread_A_B(A x,B y)
{
int r;
r=x.a+y.b;
cout<<"sum of a and b from different class : "<<r<<endl;
}

 main()
{
A obj1;
B obj2;
fread_A_B(obj1,obj2);
}
```

Friend Class

A Class all of whose member function are allowed to access the private and protected members of a particular class. The private and protected members of any class cannot be accessed from outside the class. In some situations, program may require to access these members. The use of friend's functions allows the user access these members. A function that is declared in class with friend keywords become the friend function of the class.

Code

```
#include <iostream>
using namespace std;
class A
{
private:
    int a;
public:
    A()
    {
    a=10;
    }
friend class B;
};

class B
{
public:
    B(A x)
    {
cout<<"sum of a and b from different class : "<<x.a<<endl;
    }
};
 main()
{
A obj1;
B(obj1);
}
```