Object

- ***Everything that exist, in any form is known as object such as table, fan, book, student, employee, number, air, matrix, determinant , car , tree , plant , pen and you etc.***

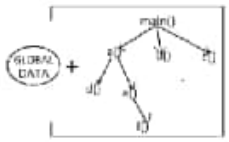
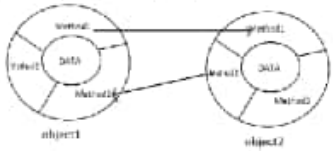
  Let us understand object by an example ***of Car***

- ***Properties***
  - Those elements that describe an object is known an properties such as in car object properties as follows
    - Model number
    - Colour
    - Price
- ***Method***
  - In c++ Any object consists of ***specified of methods*** and the user of object can use it, that means ***actions a user can perform on an object is known as method***. Such as in car object properties as follows
    - Start
    - Stop
    - Change Gear
    - Press Clutch
- ***Event***
  - When action perform the on an object, state of object get change regularly and on these different states object can trigger method means method that executed by object itself is called ***event method*** because it execute only when object comes into particular state known as ***event.***

- These characteristics form **the Object.**

***Object** Oriented Programming:*

- ***OO Application are developed** faster, easy maintainable and **secure.***
- It is a ***new programming approach or paradigm.***
- While writing an application ***objects get use.***
- In this approach all objects get communicate with each other by using their methods means there is a procedure/way to communicate so nobody can misuse the data of properties.
- ***It follow bottom - up approach.***

Difference between procedural programming and object oriented programming		
Serial No.	Procedural programming	Object Oriented programming
1	In procedural oriented approach the problem is solved by using Procedure/ functions.	In object oriented approach the problem solved by using objects
2	In procedural approach The ***Shared DATA*** is moving all around the system freely thus anybody can be misuse it easily. ***It is not secured approach.*** 	In OOPs The ***Shared DATA*** is not freely available because it is encapsulated so nobody can misuse it. 
3	In procedural approach, error detection become difficult in the respect of ***shared data***	In OOP it is easy to detect error because specific methods only access the DATA.
4	When program grow larger then procedural programming approach get fail.	Features of oops such as class, object, inheritance, polymorphism etc. are mainly used to make large or complex program.
5	For small application used procedural programming approach	OOP is for rich and secured application development.
6	Procedural programming approach does not represent real world entities directly.	Object oriented programming approach model the real world entities.

Object Based Language:

- Don't follow two concept
  - Dynamic Binding
  - ***Inheritance***

For example:

- Visual basic 6.0
- Java Script
- VB Script

Object Oriented:

- Everything is not represented in the form of object

For Example:

- ***Java***
- ***C++***

Truly Object Oriented:

- Everything expresses in the form of object with support all the features of the Opps

For Example:

Small Talk

Note: Java, C# is not truly object programming because the primitive data type are not treated as object.

But it is **perfect object oriented programming's** that balance the efficiency by making primitive data type not as objects.

CLASS

- **CLASS** describes object by defining ***properties, method and may consists of signals.***
- This ***blueprint*** or ***design*** of ar object is called ***class***
- In C++ technology there more than 5000 classes already exist such as ***IOS***, ***ofsteam***, ***ifstream, fstream , isteam , ostream, iostream, istream_withassign, ostream_withassign etc.***
- We can create blueprint in C++ technology there is keyword ***class/struct*** .
  - **Note**
    - ***It highly recommended using class keyword to create blueprint of the object.***
  - ***What is the difference between class and struct(ure) in C++***
    - ***All the member of class by default has private whereas structure has public.***

Syntax:

```
class <ClassName>
{
        Properties;
        Methods ;
};
```

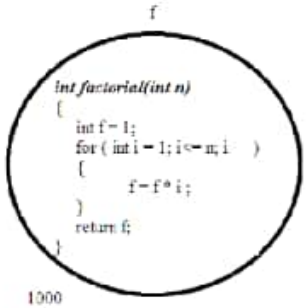
- ***Any element that defined in class known as member.***

e.g.

```
/*
   Define  a class Myfactorial having a public member function ( method )  factorial that has a
   parameter (n) of type int and f nd out the factorial of n,  then return factorial
*/
class MyFactorial
{
   public:
   int factorial(int n)   <- Member function
   {
           int f = 1;
           for ( int i = 1; i<= n; i++ )
           {
                   f = f * i ;
           }
           return f;
   }

};
```

- Once the ***class / blueprint is formed,*** *we can* ***create any number object of its type and then we can access their members.***

- ***Syntax***
  [class] ClassName ObjectName;
- ***e.g.***
  class MyFactorial f;
  MyFactorial f;



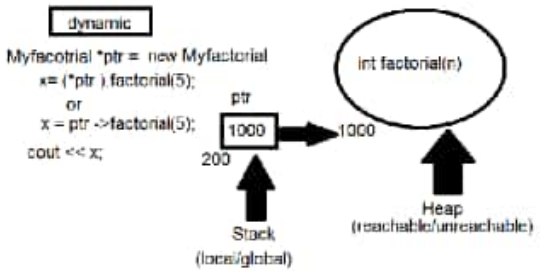
Using statically object

```
#include<iostream>
using namespace std;
class Myfactorial
{
        public:
                int factorial(int n)
                {
                        int i , f = 1;
                        for( i = 1 ; i<= n ; i++ )
                        {
                                f = f * i ;
                        }
                        return f;
                }
};

main()
{
        Myfactorial f;
        cout << "enter number";
        int num ;
        cin >> num;
        int x = f.factorial(num);
        cout << "factorial is " << x ;

}
```

using dynamic object

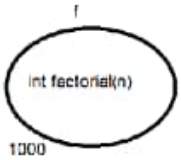
```
#include<iostream>
using namespace std;
class Myfactorial
{
        public:
                int factorial(int n)
                {
                        int i , f = 1;
                        for( i = 1 ; i<= n ; i++ )
                        {
                                f = f * i ;
                        }
                        return f;
                }
};

main()
{
        Myfactorial *ptr = new Myfactorial;
        cout << "enter number";
        int num ;
        cin >> num;
        //int x = (*ptr).factorial(num);
        int x = ptr->factorial(num);
        cout << "factorial is " << x ;

}
```

Fig



Statically

```
MyFactorial f;
x = f.factorial(5);
cout << x ;
```

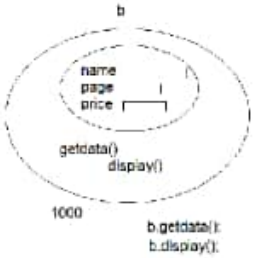


```
e.g.
#include "CFactorial.cpp"
#include <iostream>
using namespace std;
main()
{
    int n, r;
    float ncr;
    cout << endl << "Enter value of n";
    cin >> n;
    cout << endl << "Enter value of r";
    cin >> r;
    Myfactorial a;
    ncr = a.factorial(n) / (a.factorial(r) * a.factorial(n-r));
    cout << "ncr value is " << endl << ncr;

}
```

- we can access members of object using dot (.) operator
  - object.member
  - e.g.
    - f.factorial

```
/*Create a class Book consist of following properties and  methods(member
functions)
Properties: it hold the information of book name, page, price
Methods  :
getdata() method that take name, page and price
                and store into the properties of
                object
display()
                method print the properties of object.
*/
    #include <iostream>
    using namespace std;
    class Book
    {
        private:
            char name[20];
            int page;
            float price;
        public:
            void getdata()
            {
                cout << endl << "Enter book name";
```

```
                //cin >> name ; // cin.operator(name)
                cin.getline(name,20);
                cout << endl << "Enter pages ";
                cin >> page;
                cout << endl << "Enter price ";
                cin >> price;
        }
        void display()
        {
                cout << endl << "Book #";
                cout << endl << "Name : " << name;
                cout << endl << "page : " << page;
                cout << endl << "price: " << price;
        }
};

main()
{
        book b;
        b.getdata();
        b.display();

}
```



Note:

* By default all the members if the class are ***private***.

e.g.

```
#include <iostream>
using namespace std;
class Book
{
        char name[20];
        int page;
        float price;
    public:
        void getdata()
        {
                cout << endl << "Enter book name";
                //cin >> name ; // cin.operator(name)
                cin.getline(name,20);
                cout << endl << "Enter pages ";
                cin >> page;
                cout << endl << "Enter price ";
                cin >> price;
        }
```

Private member by default

```
        void display()
        {
                cout << endl << "Book #";
                cout << endl << "Name : " << name;
                cout << endl << "page : " << page;
                cout << endl << "price: " << price;
        }
};
```