→ It is not a language, it's an approach to write code

Object Oriented Programing

C++ language was first designed with the main intensions of adding object-oriented feature in C, at that time it is called 'C with classes'.

The main motive to use oop is it makes ~~read~~ it easy to read & understand the code, it makes easy to maintain the code, also it increases bug-free nature of the code.

When size of code increases in languages like 'C' (procedual programming language) it decreases readability, maintainability and bug-free nature of code. As they keep in calling functions or relied on procedures.

By using classes in c++ we can solve all these problems.

Also c++ gives us ability to secure data & variables by using class and object. In c++ all functions can't directly have acess to all variables so it easy to manage & secure data.

What is class :-

We have learnt earlier that class is a datatype (user defined), beside that class is templet in which we declare our functions and variables. We use them (functions & variables) through object in main() function.

Also ~~class~~ you can say that Class is encapsulate functions & variables.

Also, we need object because variables & function we have declared in class, we can't run them ~~as actully they are haven't~~ due to the fact that they didn't aquir space in memory unless we make object for them in main().

smallest/basic
Object are those ^ entities which being run in OOPs.

We can make different object through them we & ^ can run the same coade as many times as ~~we b~~ many object we have.

So, it's easy and we don't need to write same coad many times, we just make another object which uses same coad & classes but runs seprately from other object ~~and differ~~
~~object never interfear~~

Also if we have two different object for same class, these two object will use same class but never interfear in their work flow or data uses.

If all these seems you head load, you can understand it as class is a datatype in which we have defined different variables & functions but if we want to use them we make object (For demonstration or understanding sake we can think an object as variable, made with that user defined data type).

Benefits of OOPs :-

→ Better code reusability using objects and inheritence.

→ principle of data hiding and data security helps build ~~secu~~ secure system.

→ Software complexity can be easily managed.

→ Multiple objects can co-exist without any interference.

→ Syntex of class and access modifiers in class

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

Class x
{
```

```
private :
    //variables and function
    int a,b;
Public :
    //variables and functions
    int c;
}; void write();

void x :: write()
{
  cout<<"Hellow world"<<endl;
}

int main()
{
  x world;

  world.write();

  return 0;
}
```

object

In class there are 3 type of access modifiers (privat, public and protected) but we generally use two of them (privat & public)

By using access modifiers we can control sharing and access of data, which makes our system secure.

If we place a variable in privat class it can't be accessed directly from main by any means, it can only be accessed by member functions of same class.

If we put variables and functions in public class then we can access them directly from main by use of object of same class.

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

class mark
{
  private:
    int math, sci, eng;
    int total;
  Public:
    int hindi;
    int input(int a, int b); int();
    int sum
    int input();
    void sum();
};

int mark :: input()
{
    cout<<" put your marks in 3
          subjects "<<endl;

    cin>> math >> sci >> eng;

}
```

```
Void mark :: sum()
{
    total= math+ sci + Eng;

    cout<<" Total is"<<total<<endl;
    cout<<" marks in hindi is"<<hindi;
}

int main()
{

    mark akash;
    akash.hindi=95;
    akash.input();
    akash.sum();
    akash.hindi=100;
    return 0;
}
```

//Here in main function we have directly access hindi variable but we can't access ~~variable~~ variable like math in this manner as math, sci & Eng are private variable. We can only use them via member function of same class.

NOTE:- You can declair object along with the declaration of class for example.

```
class    X
{
    // Variable and function
} Harrish, Ramesh ;
```

here, Harrish and Ramesh are object.

As we have studied earlier we can make functions and variables public. To use them we need member function of same class.

If we have a private function we ~~can't~~ will use that function via another function, so, in this condition we will do Nesting of function. We can

do nesting of functions wheather they are private or public but if a function is private we can only access that function by nesting it in another member function.

ex. -

```
#include<iostream>
#include<string>

using namespace std;

class binary

{

    string s;
  void chk_bin(void);
  Public:
    void input();
    void one_comp();
};
```

```
void binary :: input()
{
    cout<<"Input a binary number"
        <<endl;
    cin>>s;
}

void binary :: chk_bin(void)
{
    for(int i=0;i<s.length();i++)
    {
        if(s.at(i)!=0 && s.at(i)!=1)
        {
            cout<<"error not binary"<<endl;
            exit(0);
        }
    }
}

void binary :: one_comp()
{
    chk_bin();
```

```
for(int i=0; i<S.length() ; i++)
{
  if(S.at(i) == 1)
  {
    cout<<" 0";
  }
  else
  {
    Cout <<" 1";
  }
 }
}

int main()
{
  binary check;
  check. input ();
  check. one_(omp();
  return 0;
}
```

NOTE:-

In this programme

S.length() is equivalent to length(S);

and

S.at(i) is equivalent to S[i];

also

exit() function is used to stop the running load and come out of programme.

Here, we can't call chk_bin() function directly in main() as it is private function, so, we have nested it in one_comp() function

→ Memory allocation while making Object and class.

We all know that when we make class and declair function and variable then memory is not allocated at that time. Memory is allocated when we make object through those classes.

If we have one class and we have made 1000 of object for that class then memory for all variables and function will also be allocated 1000 times. It will cause problem for memory.

So, our compilar intelligently allocate memory. If any function or variable is only needed allocating memory once (variable or function which are common for all objects) it will allocate them

only onece, if some other func or variables needed allocating memory each for each objects it will do so automatically.

→ Use of static variable in class:-

We all know that static variable is a special variable which only initialize once, even if it is in loop or recursion. so it can be helpful to store data like counting no. of something or some important data.

Now, declaration of static variable in class happen/occurs in different manner as if we define it as normal variable in class, and we made many objects then it should be initialize as many times as many are the objects can't be possible for a static variable.

So, static variable is class is not associated with any object but with the class itself, so it is defined outside the class.

Let's take an example question to demonstrate it.

We have to make a programm to take input personal information of employee and count their no. using class.

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

class employee
{
    int Id;
    char name[20];
   static int count;
  public:
     void getdata(void);
     void printdata(void);
};

void employee :: getdata()
{
     cout<<" Input employee id"<<endl;
     cin>>Id;
```

```
cin>>cout<<" Input name "<<endl;
    cin>> name;
    count ++;
}

Void Employee :: printdata()
{
    cout<<" Empolyee id is "<< Id <<end
    cout<<" Employee name is "<< name <<end
}   cout<<" count ";

int Employee :: count;

// By doing so we have made
   a static variable count of type
   int which is only/directly
   related to class not with
   objects.

int main()
{
   Employee Raj, Reshma, Sonam;
   Raj . getdata();
   Raj . printdata();
```

```
Reshma.getdata();
Reshma.printdata();

Sonam.getdata();
Sonam.printdata();

return 0;
}
```

Q. WAP for shop to take input item id and item price and print it.

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

class shop
{
    int id;
    int price;
    static int count; int p;
public:
    void counter(void)
    {
        count=0;
    }
```

```
void getdata(void)
void printdata(void)
};

void shop::getdata()
{
    cout<<" write item id" <<endl;
    cin>> id;
    cout<<" write item price" <<endl;
    cin>> price;
    count ++;
}

void shop :: printdata()
{
   cout<<" input item no whose
          id and price you
          want to see" <<endl;
   cin>> p;

   if( p <=count )
   {
     for(int i=0; i<count; i++)
     {
        if( i == (p-1))
        {
```

```
cout<< count;
cout<<" item id is "<< id <<endl;
cout<<" item price is "<< price <<endl;
cout<<endl<<endl;
}
}
}

else
{
    cout<<" Given no is
        wrong, there is no
        item with this no"<<endl;
}
}

int shop :: count;

int main()
{
    int P;
    cout<<" How many items you want
        to put in"<<endl;
    cin>> P;
```

```
shap first;
first.counter();
for(int i=0; i<p; i++)
 {
    first.getdata();
 }

first.printdata();
return 0;
}
```

Also note that memory allocation of static variable happens only once as it is only initialize once. Also memory allocation is associated with class not object for static variable.

Like normal variable, if we want to initilize it from 10 we write

```
int x=10;
```

in class to initilize as static variable we ~~can't~~ can't initialize it in class, we have to initilize it outside

static variables are also called class variable as for every class they are shared. We can also access them like, {{class name}} :: static var;

class as and {{object name}}.static var; but first one is recommended.

int <class name> :: y = 10;

Also note that by default static variable has value of '0' initialize in it.

→ Static function :-

~~Static member function are made in class~~ can

static member function only access static data member and other static function, they can't access other private data members and functions.

Also static function can be called without making object. Directly with class. like

{{class name}} :: static func();

We can't use this pointer with static func