

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

Object Oriented Programming

C++ language was first designed with the main intention 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 language like 'C' (procedural 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 access 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 template 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 actually they ~~were~~ haven't due to the fact that they didn't ~~occupy~~ ~~occupy~~ space in memory unless we make object for them in main().

Object are those entities which being run in OOPS.

We can make different object through them we can run the same code as many times as we have many object we have.

So, it's easy and we don't need to write same code many times, we just make another object which uses same code & classes but runs separately from other object and different object never interact.



Also if we have two different Object ~~for~~ for some class, these two object will use same class but never interfere in their work flow or data uses:

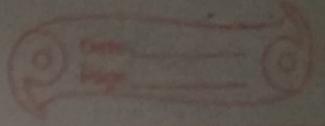
If all these seems you head load, you can understand it as class is a datatype in which we have defined different variables & functions but if we want to use them we make object (For demonstration or understanding sake we can think ~~as~~ an object as variable, made with that user defined data type).

Benefits of OOP's :-

- Better code reusability using objects and inheritance.
- principle of data hiding and data security helps build ~~seg~~ secure system.
- Software complexity can be easily managed.
- Multiple objects can co-exist without any interference.
- Syntax 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 << "Hello world" << endl;

int main()

{

x world;

object

world.write();

return 0;

}

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

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

If we place a variable in private 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 b(); int c();

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;

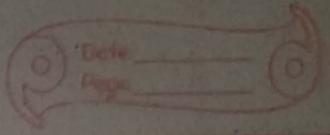
cout << "Total is " << total << endl;
cout << "marks in Hindi is " << hindi;

int main()

mark akash;
akash.hindi = 65;
akash.input();
akash.sum();
~~akash.hindi = 65;~~

return 0;

Here in main function we have
directly access ~~hindi~~ variable
but we can't access ~~math~~ 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 declare 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

Date _____
Page _____

do nesting of functions whether 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_compl();
```

```
y;
```

void binary :: input()
{

cout << "Input a binary number"
<< endl;

cin >> s;

}

void binary :: ch~~k~~_bin (void)

{

for (int i = 0; i < s.length(); i++)

{

if (s.at(i) != '0' && s.at(i) != '1')

{

<endl

cout << "Error not binary";
exit(0);

3
3

void binary :: one_comp()

{

ch~~k~~_bin();

~~ANS~~

for (int i=0; i < s.length(); i++)

{

if (s[i] == 1)

{

cout << "0";

}

else

{

cout << "1";

}

}

}

int main()

{

binary check;

check::input();

check::one_compl();

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 code
and come out of progra-
mme .

Here, we can't call chk**brn**comp()
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 declare function and variable then memory is not allocated at that time. Memory is allocated ~~by~~ when we make object through those classes.

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

So, our compiler 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 once, 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 initializes once, even if it is in loop or recursion. So it can be helpful to store data which is like counting no. of something or some important data.

Now, declaration of static variable in class happens/occurs in different manner as if we define it as normal variable in class, and we made many objects then it should be initialized 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 program 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;
~~cout~~ cin >> Id;

cin>>cout<< "Input name " << endl;
cin>> name;
Count ++;

3. void Employee :: printdata()
{

cout<< Employee::id << " Id " << endl
cout<< Employee::name << " name " << endl
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;

3

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;

2

```
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))
```

```
{
```

Count << Count;

cout << " item id is " << item.id << endl;

cout << " item price is " << item.price << endl;

cout << endl << endl;

3

3

3

else

S

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

3

3

int shop :: count;

int main()

S

int p;

cout << " How many items you want
to put " << endl;

cin >> p;

```
shop first;  
first::counter();  
for (int i=0; i < p; i++)  
{
```

3

```
    first::getdata();  
}
```

3

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 initialize it from 10 we write

```
int x=10;
```

In class to initialize a static variable we can't initialize it in class, we have to initialize 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 fun();`

We can't use this pointer with static function