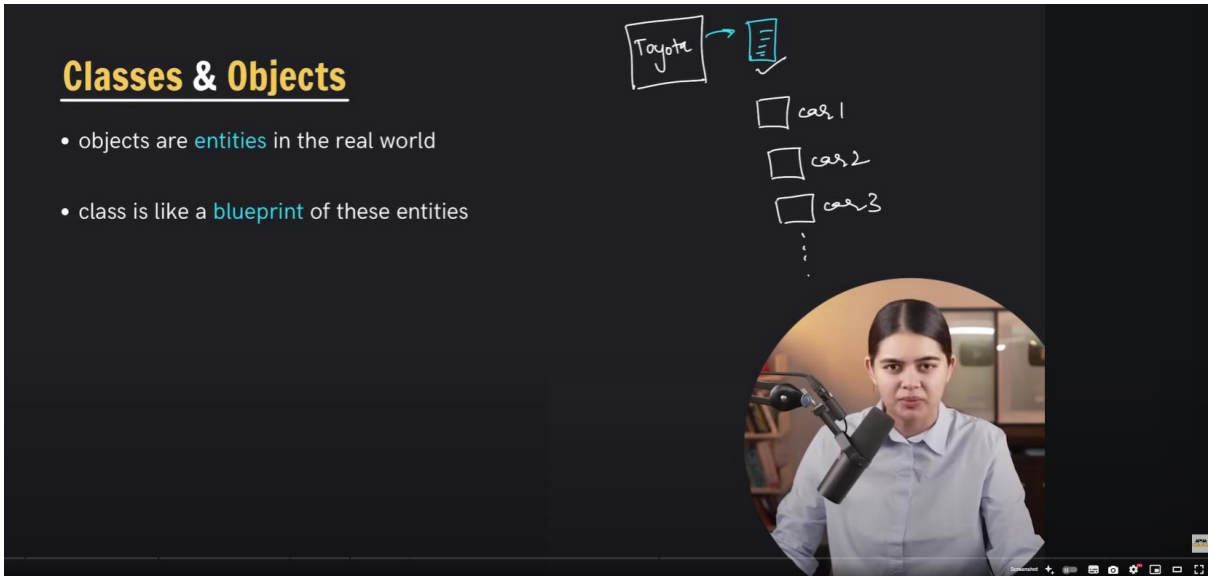
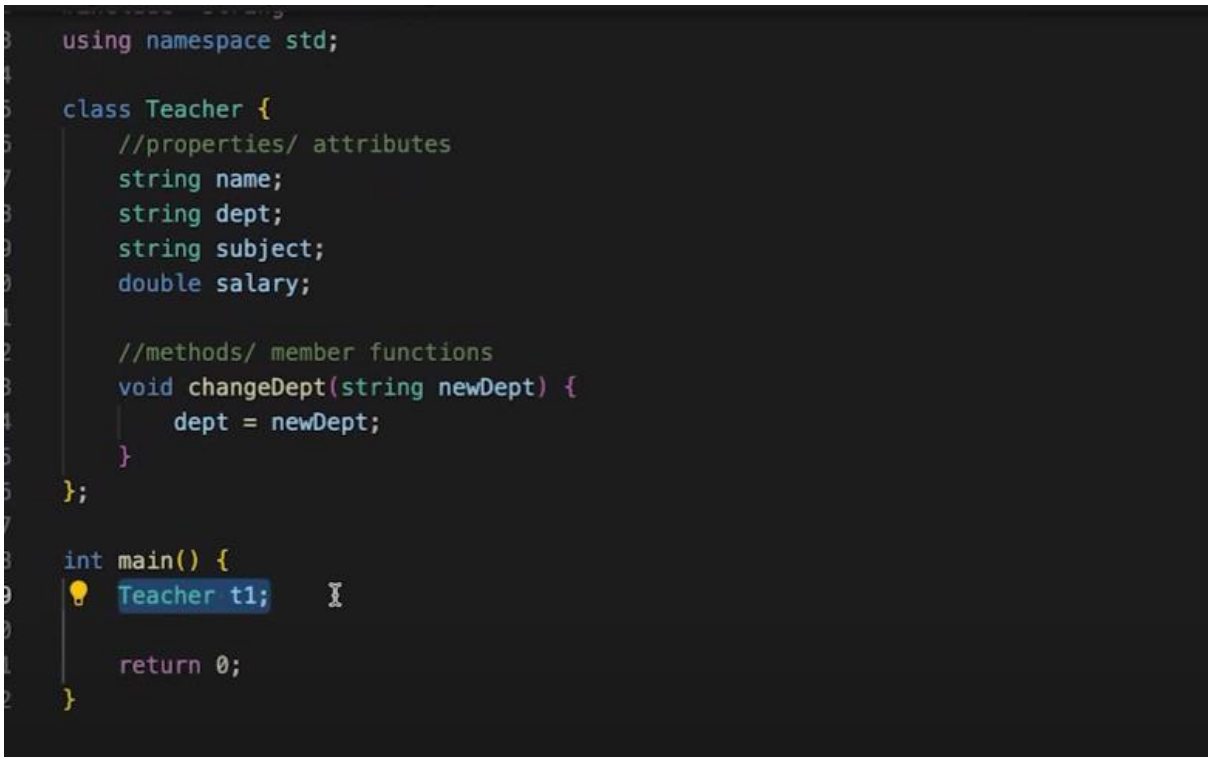
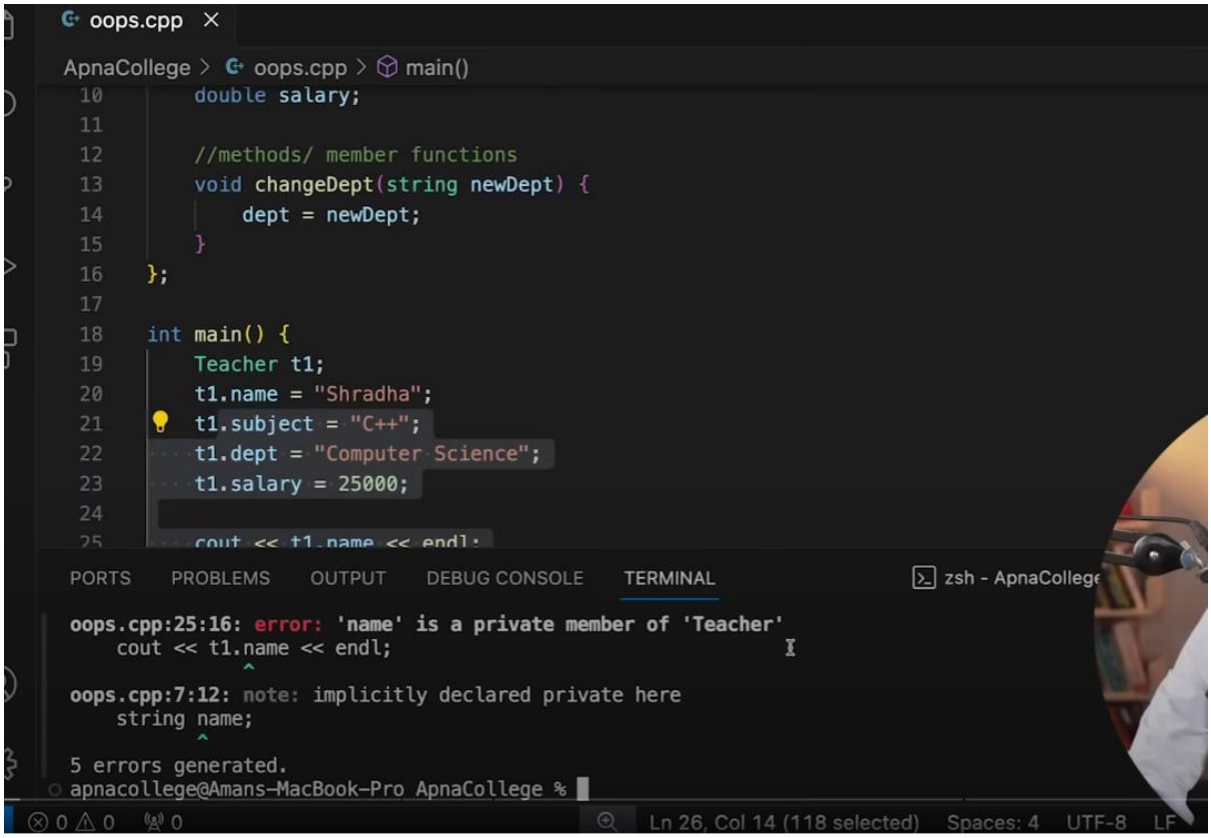
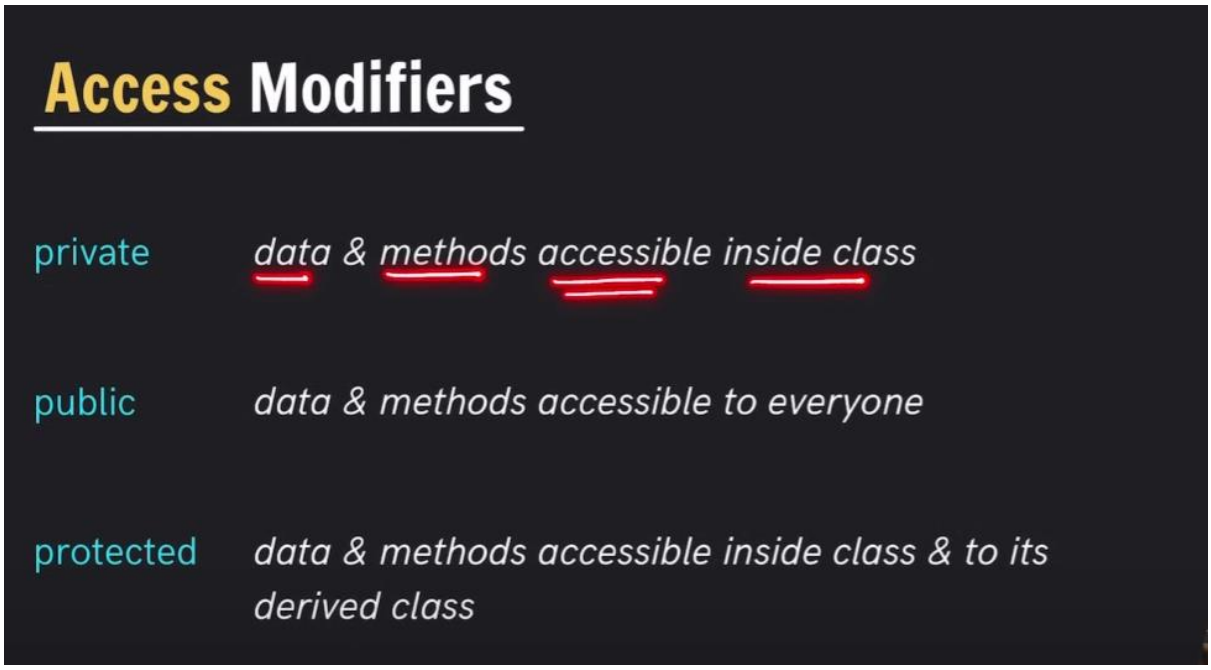


Ye wrong hai access modifier bhi dena padta hai class me







By default sab private hai

```cpp
class Teacher {
private:
    double salary;

public:
    string name;
    string dept;
    string subject;

    void changeDept(string newDept) {
        dept = newDept;
    }
};

int main() {
    Teacher t1;
    t1.name = "Shradha";
    t1.subject = "C++";
    t1.dept = "Computer Science";
```

```cpp
};

int main() {
    Teacher t1;
    t1.name = "Shradha";
    t1.subject = "C++";
    t1.dept = "Computer Science";
    t1.salary = 25000;
```

```
TS  PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

college@Amans-MacBook-Pro ApnaCollege % g++ -std=c++11 oops.cpp && ./a.out
.cpp:24:8: error: 'salary' is a private member of 'Teacher'
t1.salary = 25000;
   ^
.cpp:7:12: note: declared private here
double salary;
       ^
ror generated.
college@Amans-MacBook-Pro ApnaCollege %
```

3 / 2:04:22 • Access Specifier › Ln 24, Col 23 (18 selected)

How to still get salary in main despite making it private

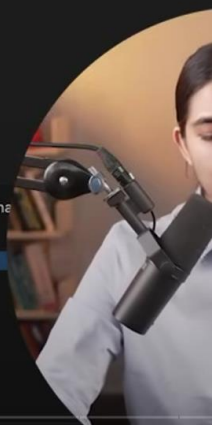
```
class Teacher {
private:
    double salary;

public:
    string name;
    string dept;
    string subject;

    void changeDept(string newDept) {
        dept = newDept;
    }

    //setter
    void setSalary(double s) {
        salary = s;
    }
    //getter
    double getSalary() {
        return salary;
    }
};

int main() {
```

```
TS  PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

acollege@Amans-MacBook-Pro ApnaCollege %
```

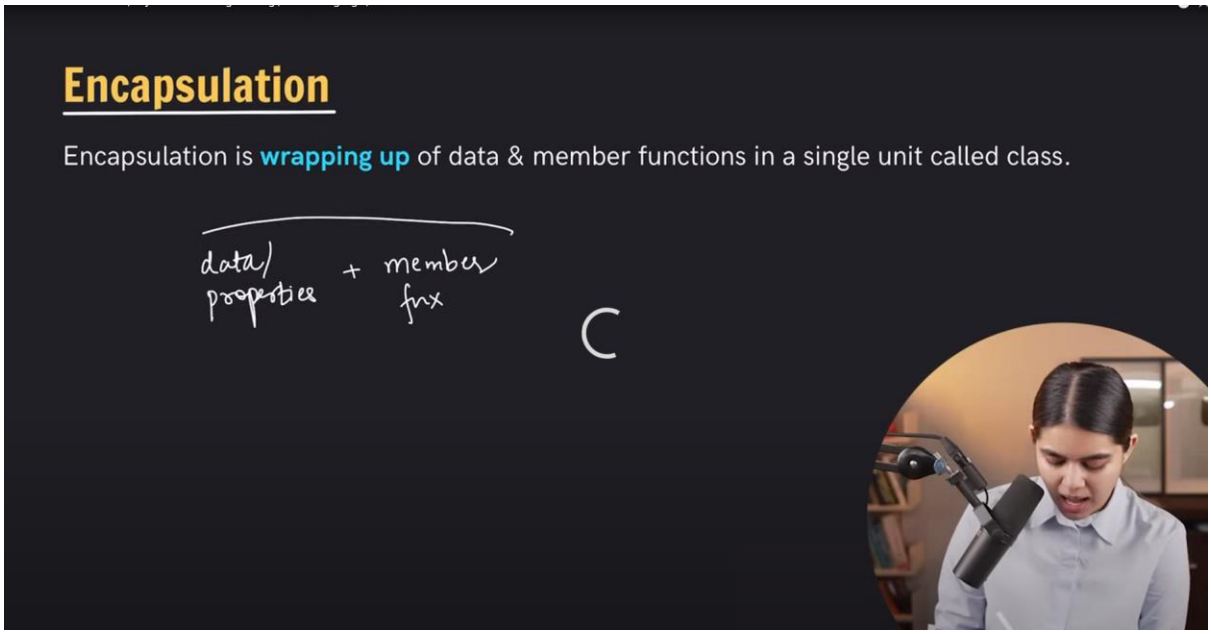
Getting salrayr indirectly through these public functions

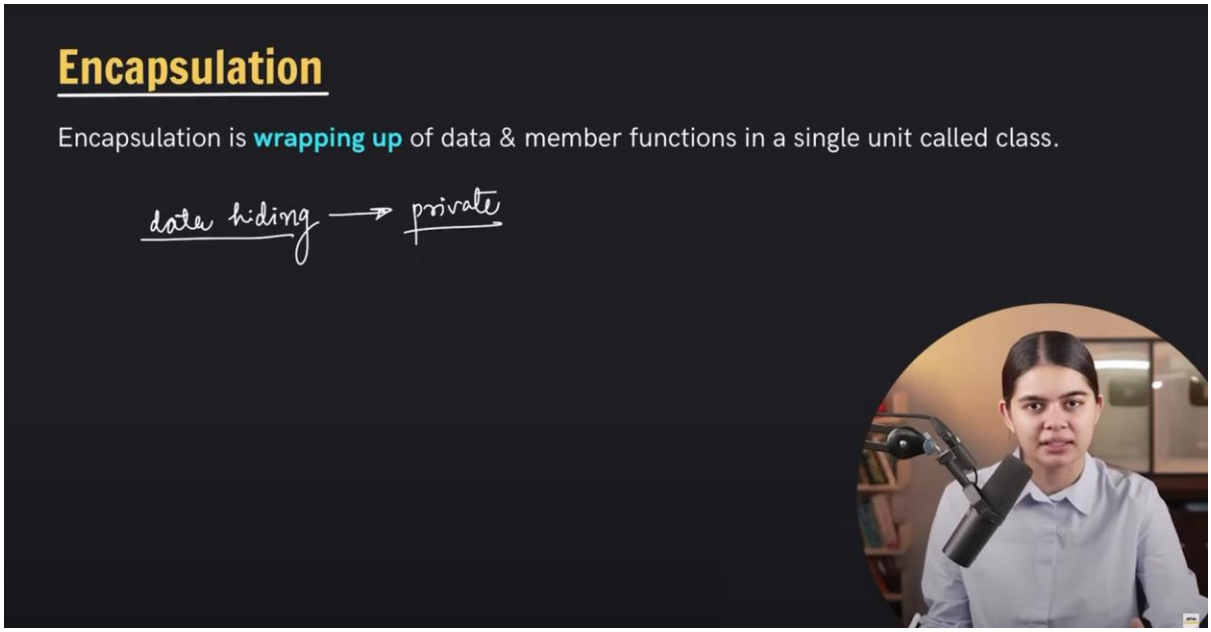
```
24          return salary;
25      }
26  };
27
28  int main() {
29      Teacher t1;
30      t1.name = "Shradha";
31      t1.subject = "C++";
32      t1.dept = "Computer Science";
33      t1.setSalary(25000);
34
```

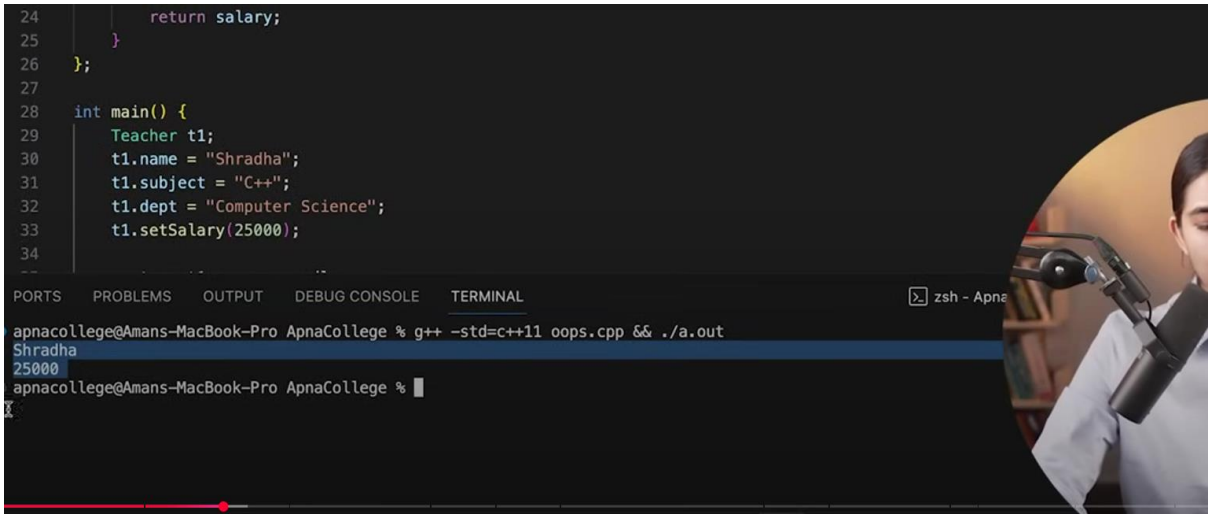
```
PORTS  PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL        zsh - Apna

apnacollege@Amans-MacBook-Pro ApnaCollege % g++ -std=c++11 oops.cpp && ./a.out
Shradha
25000
apnacollege@Amans-MacBook-Pro ApnaCollege %
```

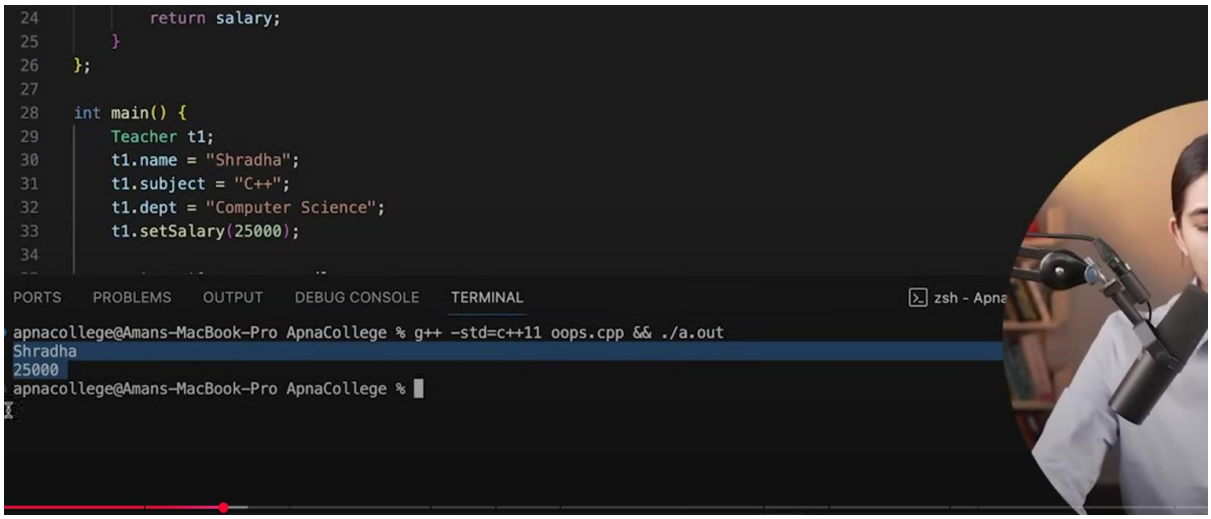


Encapsulation

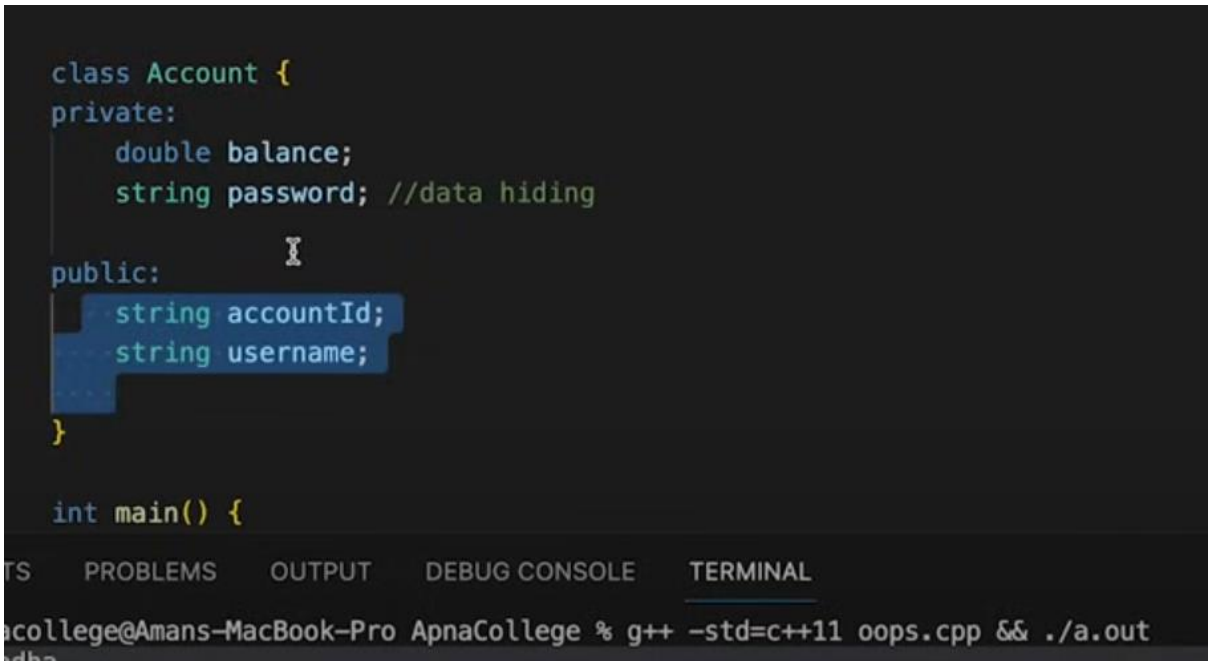




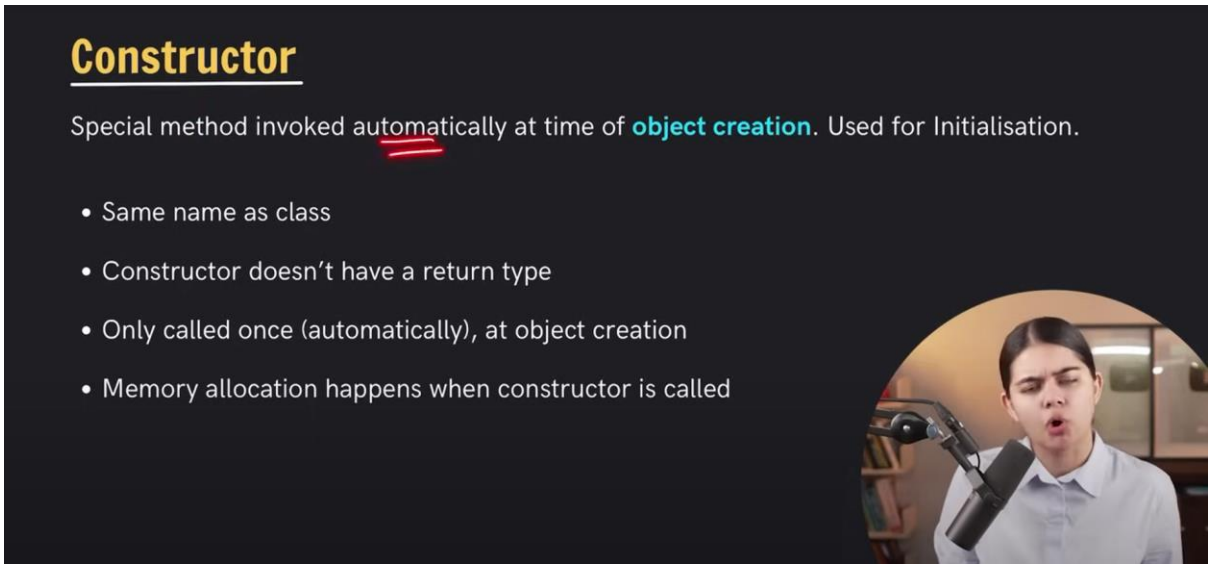


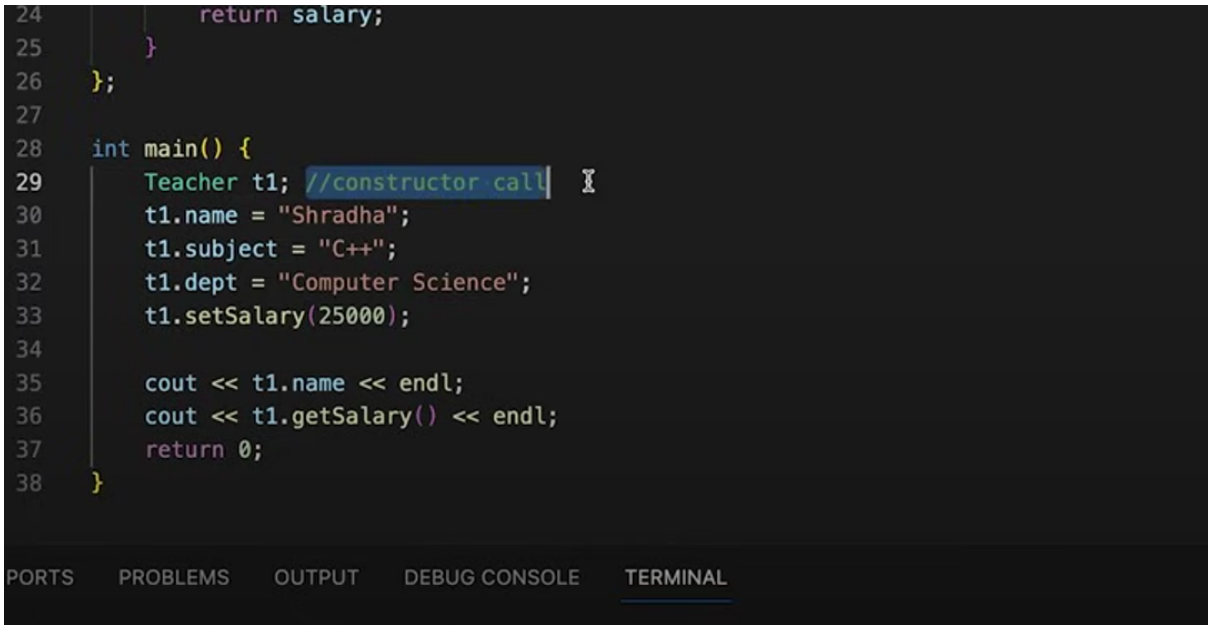


Data hiding also done using encapsulation



Constructor



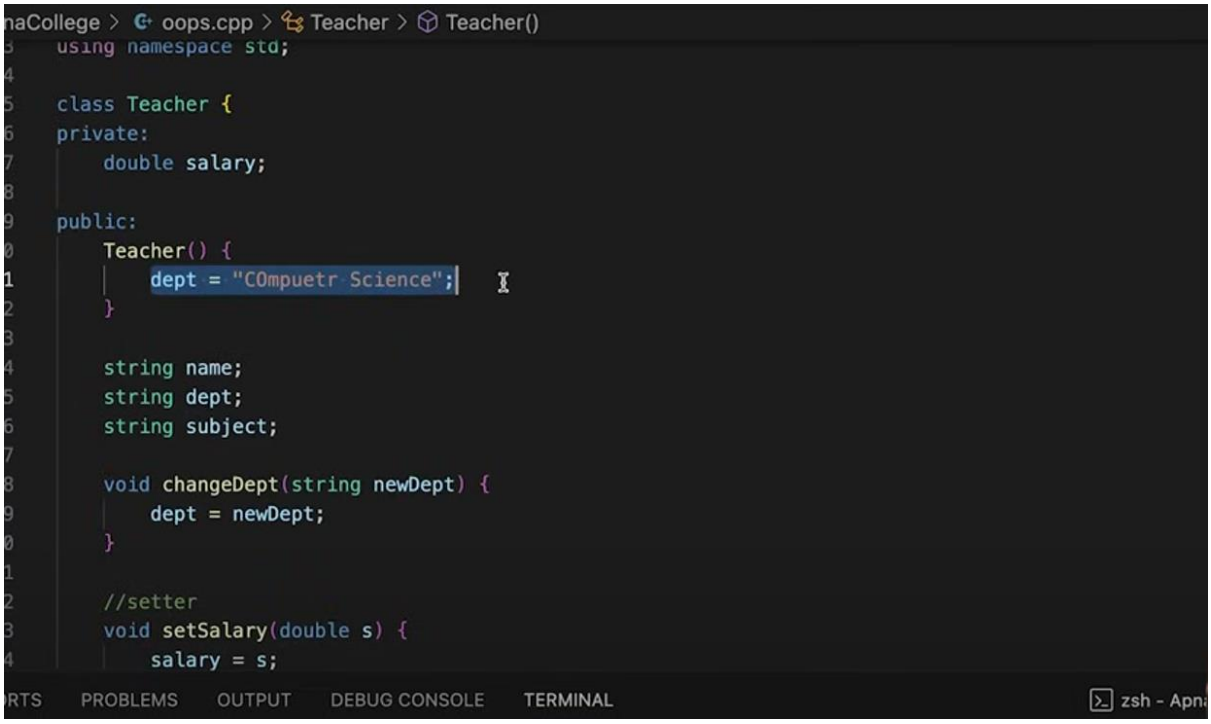


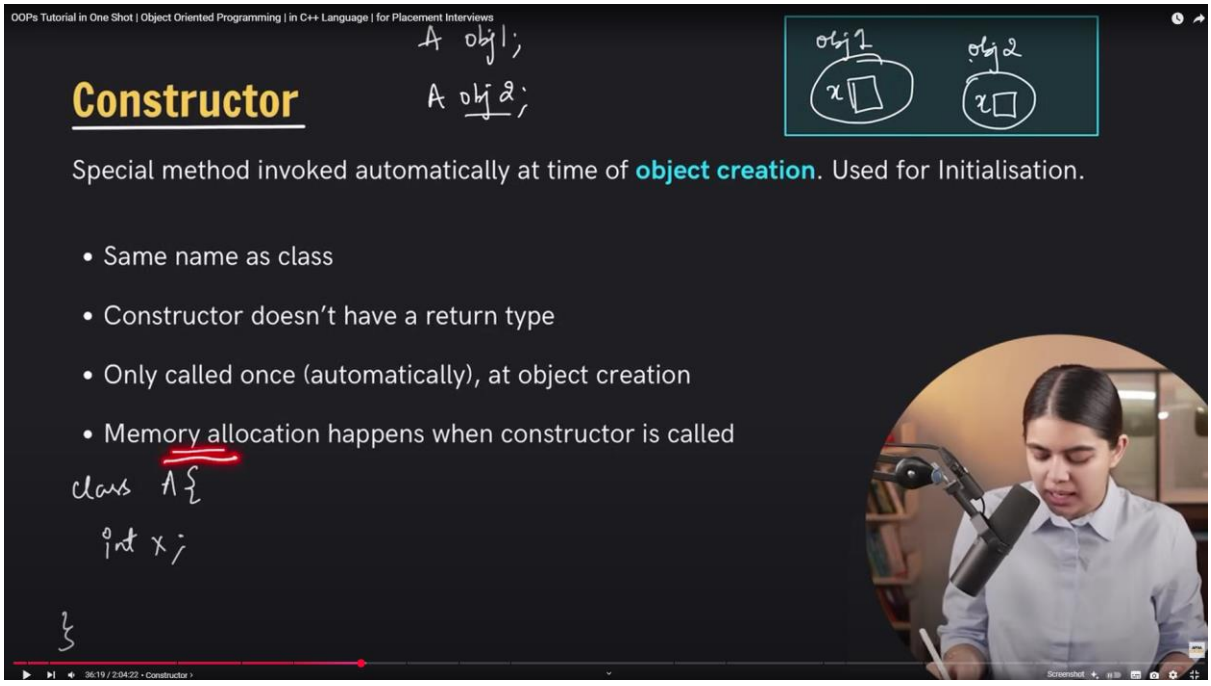
If we don't make it c++ automatically does this for us

Ya programmer constructor banata hai ya compiler

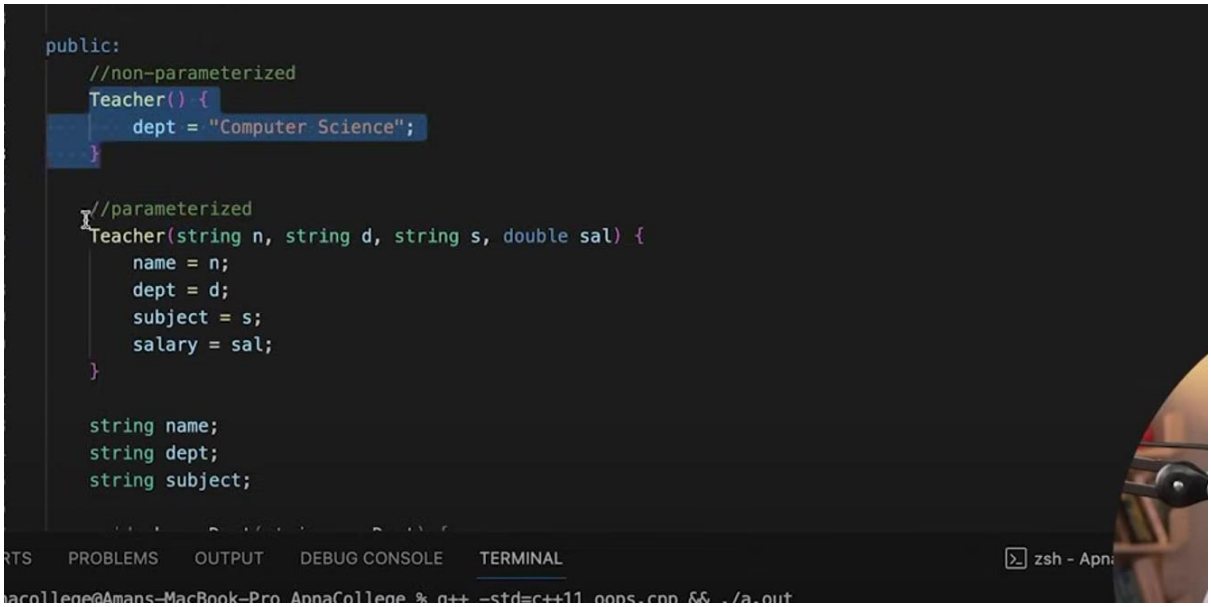
Constuctor ko Hamesha public hi banana hai

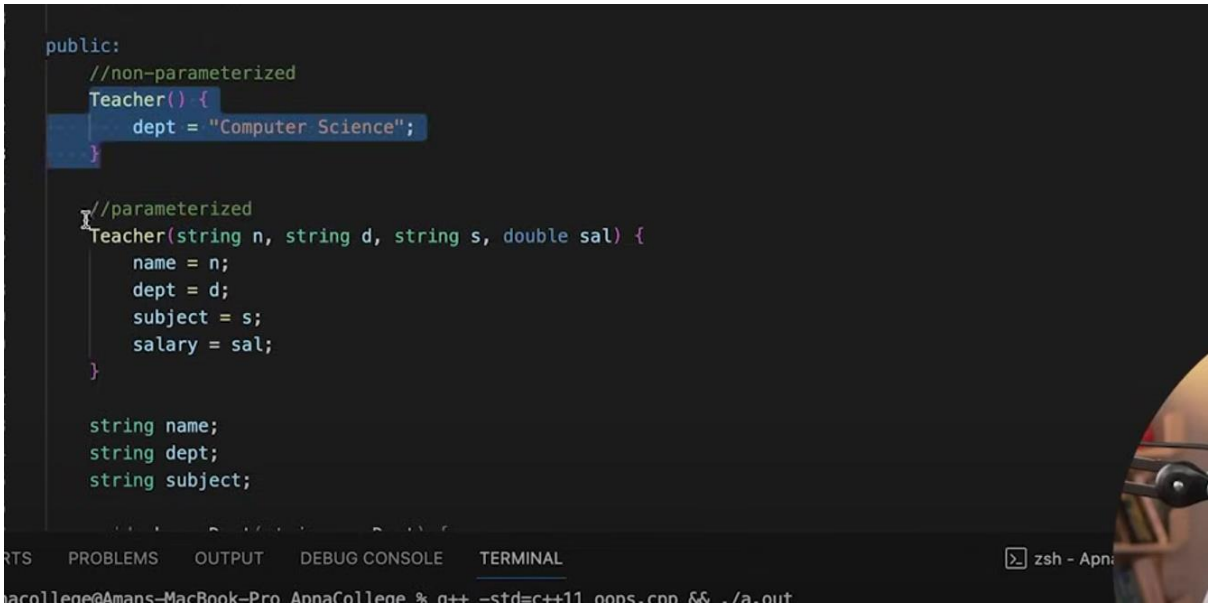
Highlighted is constructor





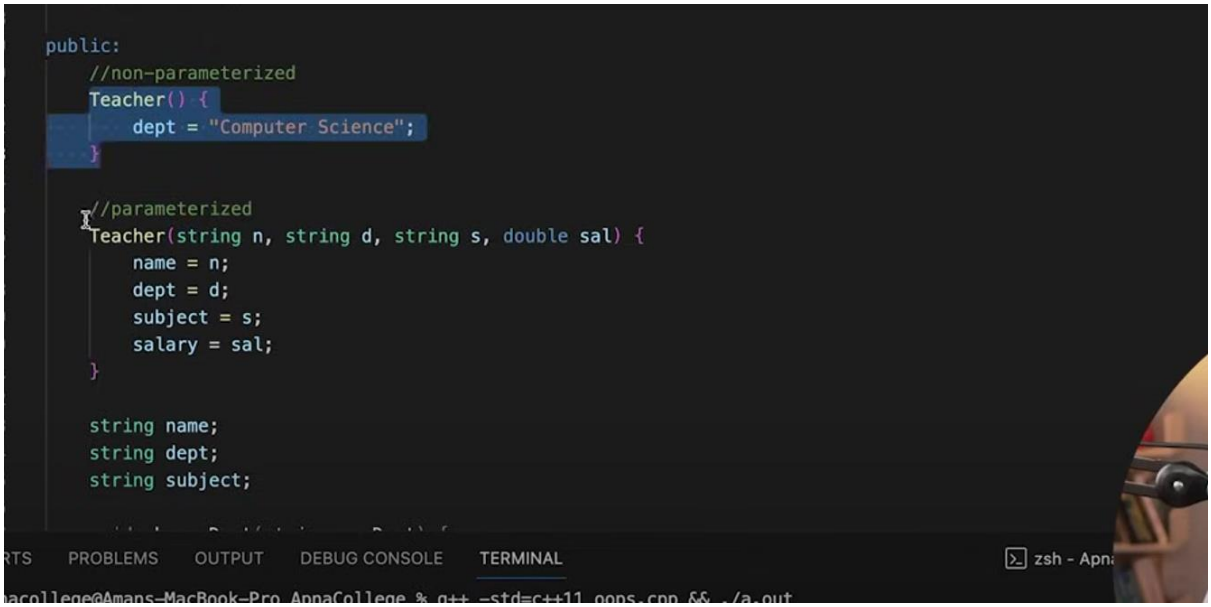
Memory allocation is for objects not classes and it happens when constructor is called





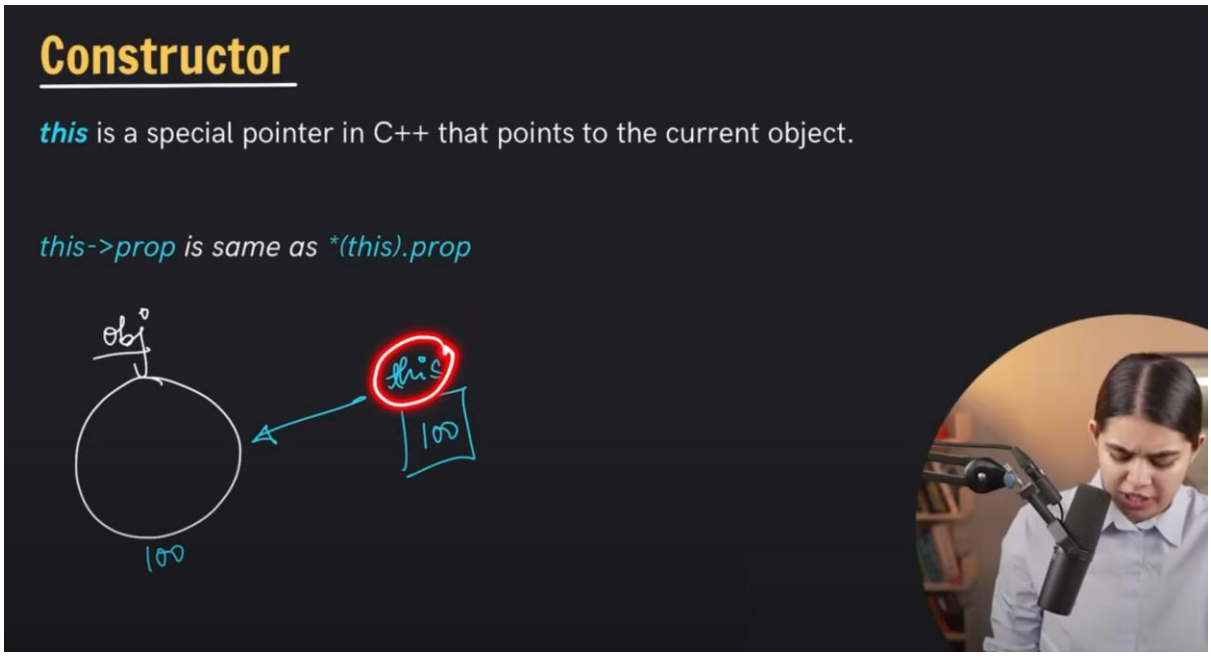
A class can have multiple constructors also given different parameters hai this is called constructor overloading

Constructor overloading is example of polymorphism



This

Current object ko point karta hai



If we want to keep parameter names similar to names of propetites in class then compiler can get comfused

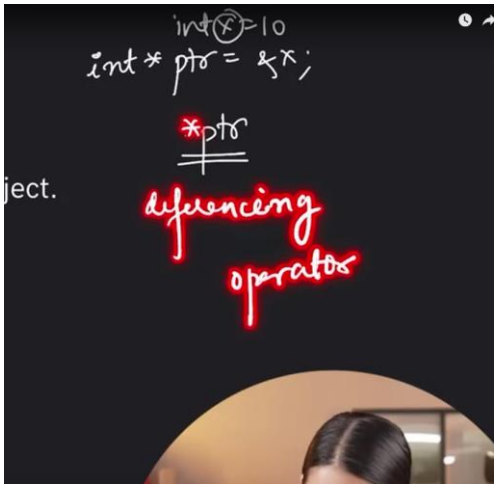
Name=name

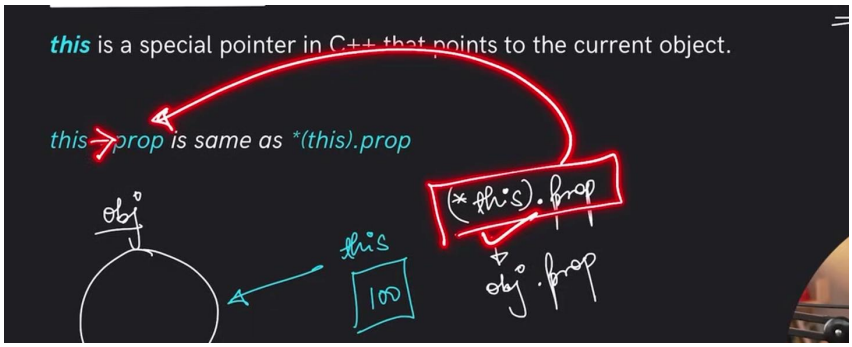
Dept=dept

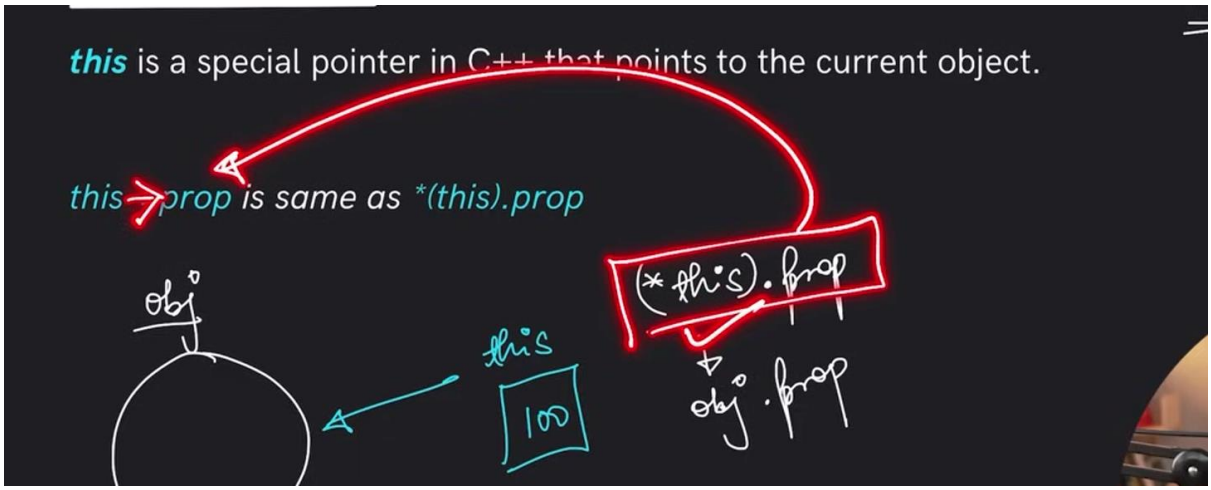
It is confusing

Left name is property of obj and right is function parameter

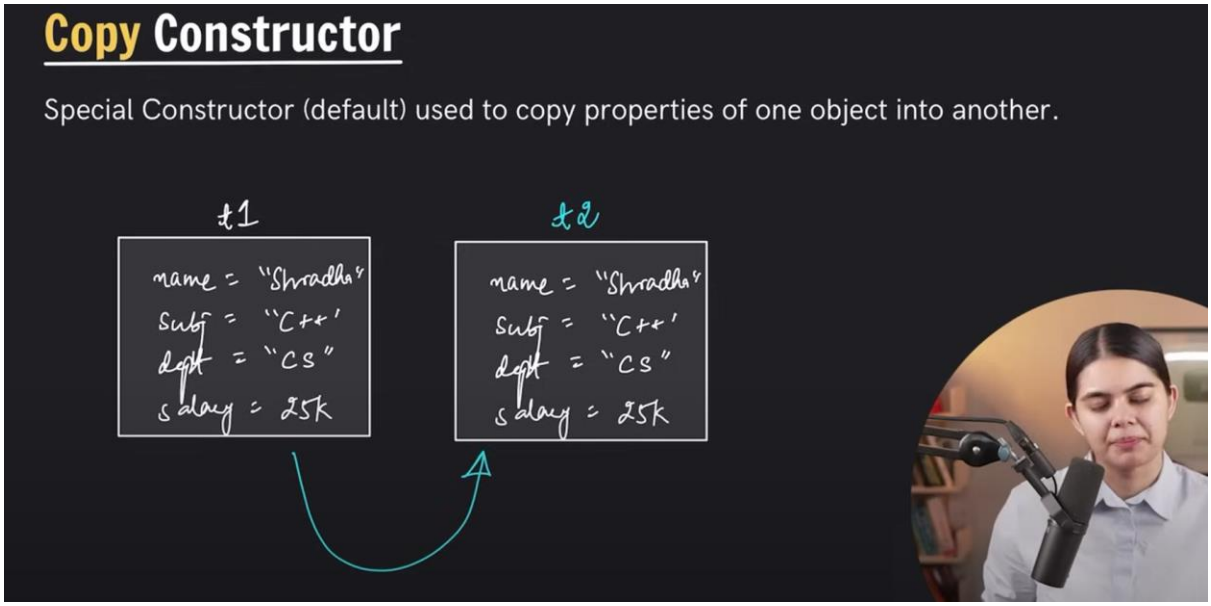
This always point to current object







Copy constructor



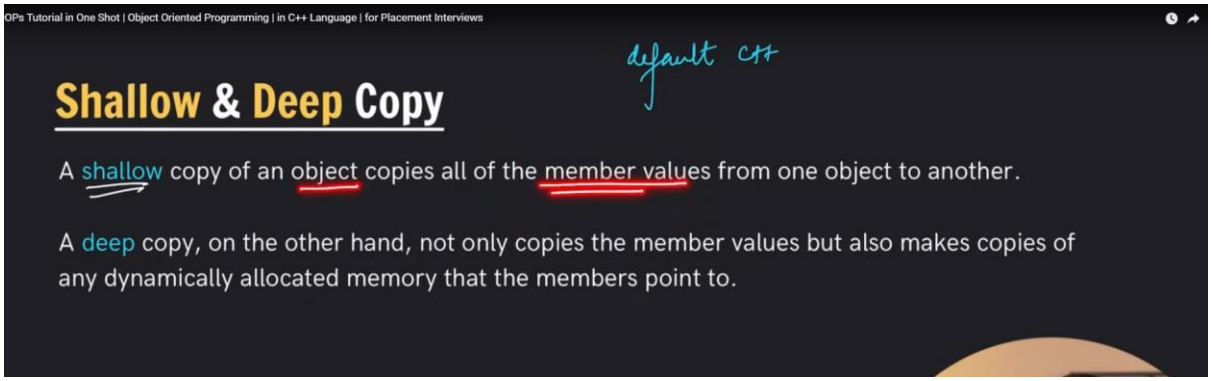

```
//copy constructor
Teacher(Teacher &obj) { //pass by reference

}

void changeDept(string newDept) {
    dept = newDept;
}
```

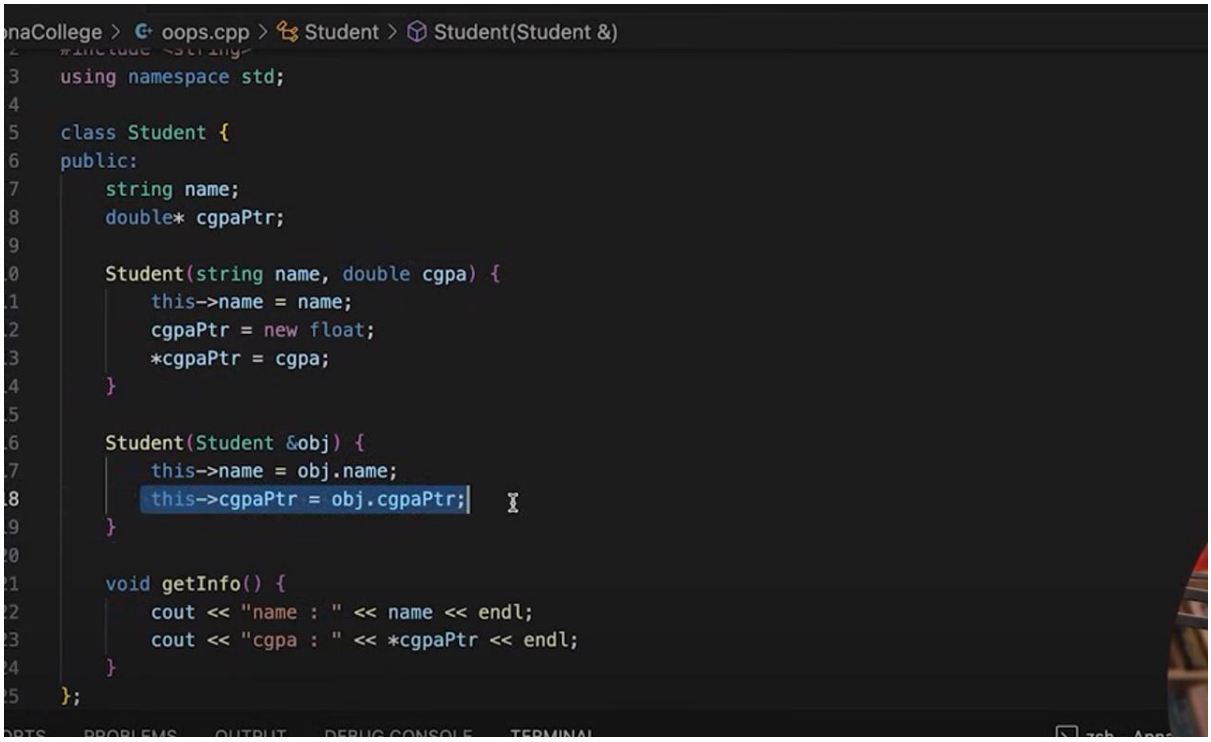
Custom copy constructor

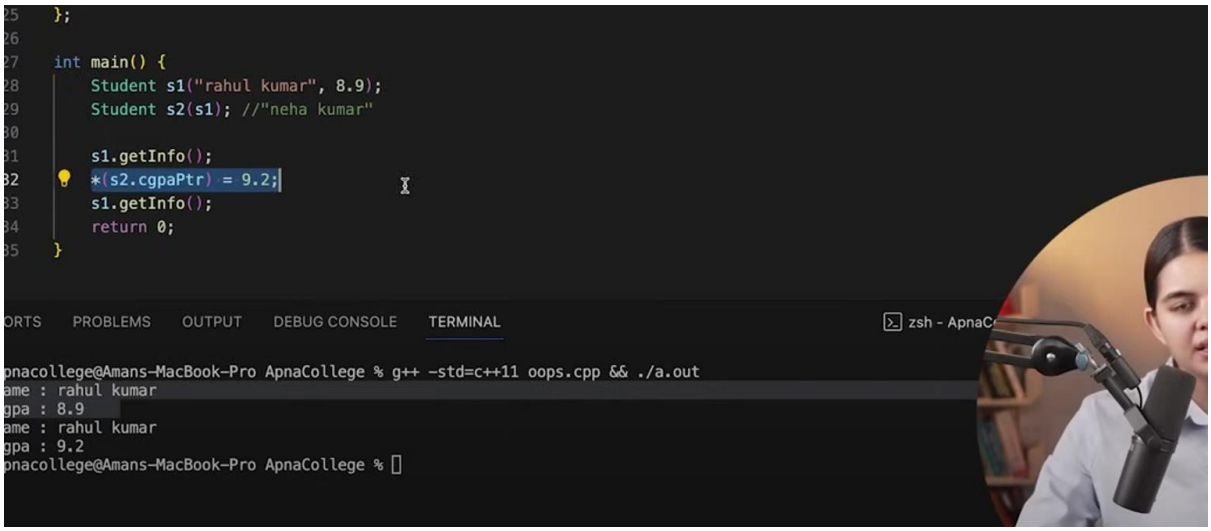
```
}

//copy constructor
Teacher(Teacher &orgObj) {
    cout << "i am custom copy constructor...\n"
    this->name = orgObj.name;
    this->dept = orgObj.dept;
    this->subject = orgObj.subject;
    this->salary = orgObj.salary;
}
```

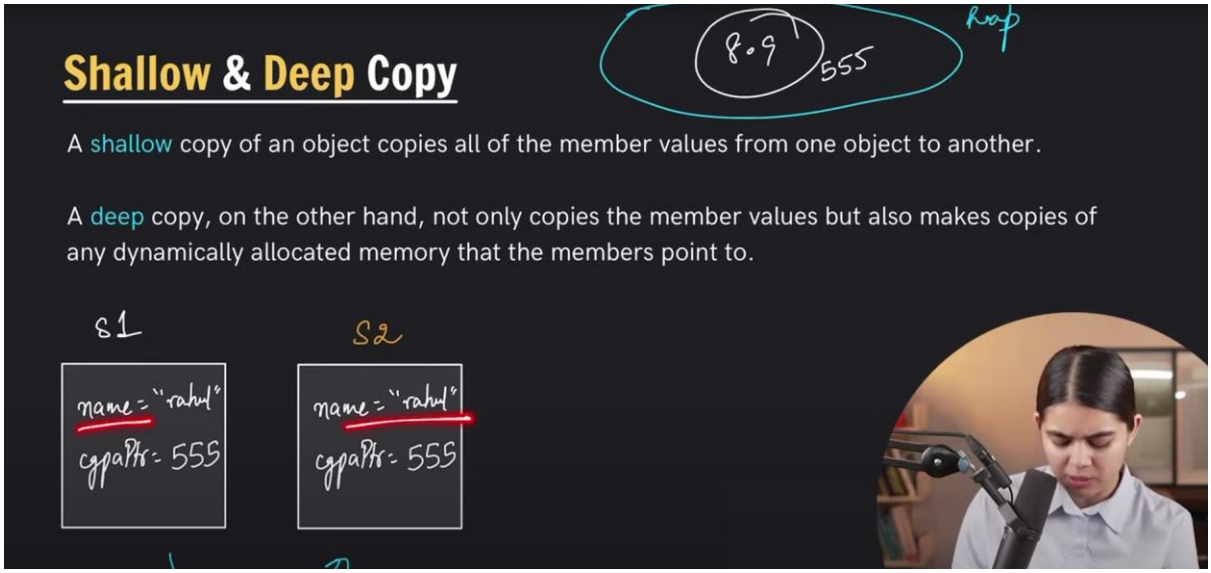

```
}

//copy constructor
Teacher(Teacher &orgObj) {
    cout << "i am custom copy constructor...\n"
    this->name = orgObj.name;
    this->dept = orgObj.dept;
    this->subject = orgObj.subject;
    this->salary = orgObj.salary;
}
```


2 types of copy shallow and deep default is shallow

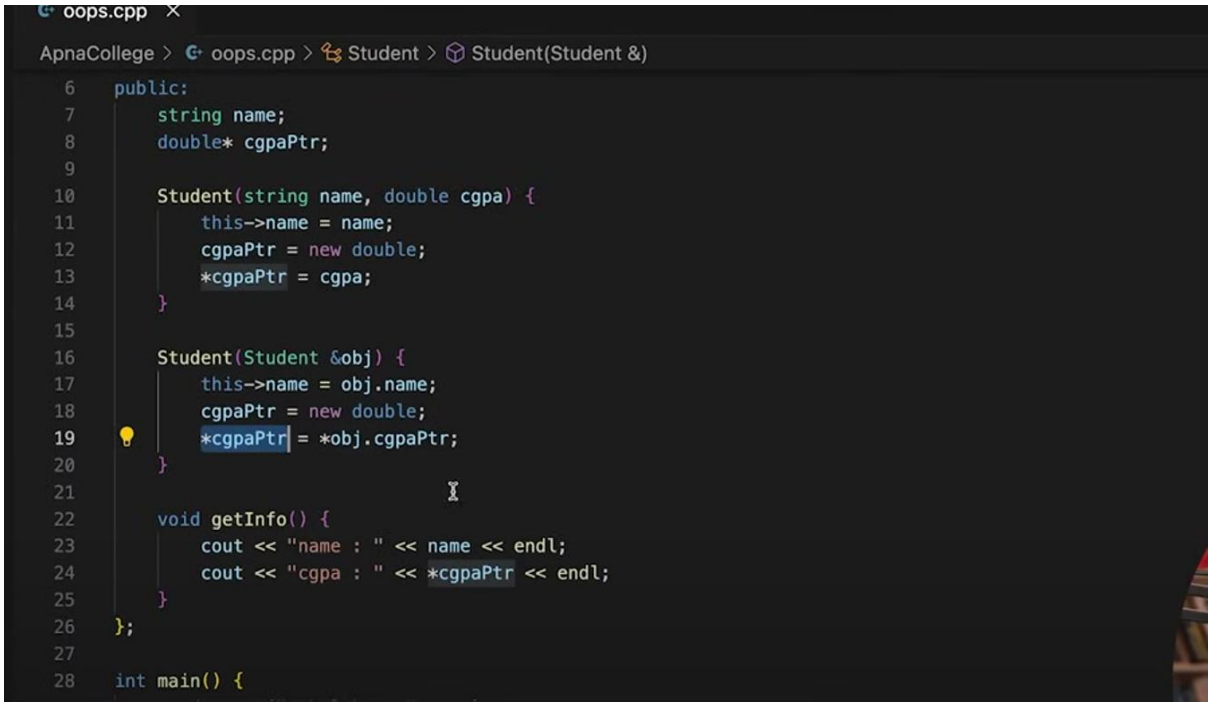



naCollege > oops.cpp > Student > Student(Student &)
using namespace std;
class Student {
public:
string name;
double* cgpaPtr;
Student(string name, double cgpa) {
this->name = name;
cgpaPtr = new float;
*cgpaPtr = cgpa;
}
Student(Student &obj) {
this->name = obj.name;
this->cgpaPtr = obj.cgpaPtr;
}
void getInfo() {
cout << "name : " << name << endl;
cout << "cgpa : " << *cgpaPtr << endl;
}
};


};
int main() {
Student s1("rahul kumar", 8.9);
Student s2(s1); //"neha kumar"
s1.getInfo();
*(s2.cgpaPtr) = 9.2;
s1.getInfo();
return 0;
}
ORTS PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
zsh - ApnaC
onacollege@Amans-MacBook-Pro ApnaCollege % g++ -std=c++11 oops.cpp && ./a.out
ame : rahul kumar
gpa : 8.9
ame : rahul kumar
gpa : 9.2
onacollege@Amans-MacBook-Pro ApnaCollege %


Shallow & Deep Copy
8.9
555
Heap
A shallow copy of an object copies all of the member values from one object to another.
A deep copy, on the other hand, not only copies the member values but also makes copies of any dynamically allocated memory that the members point to.
S1
name = "rahul"
cgpaPtr = 555
S2
name = "rahul"
cgpaPtr = 555

Deep copy


oops.cpp
ApnaCollege > oops.cpp > Student > Student(Student &)
public:
string name;
double* cgpaPtr;
Student(string name, double cgpa) {
this->name = name;
cgpaPtr = new double;
*cgpaPtr = cgpa;
}
Student(Student &obj) {
this->name = obj.name;
cgpaPtr = new double;
*cgpaPtr = *obj.cgpaPtr;
}
void getInfo() {
cout << "name : " << name << endl;
cout << "cgpa : " << *cgpaPtr << endl;
}
};
int main() {

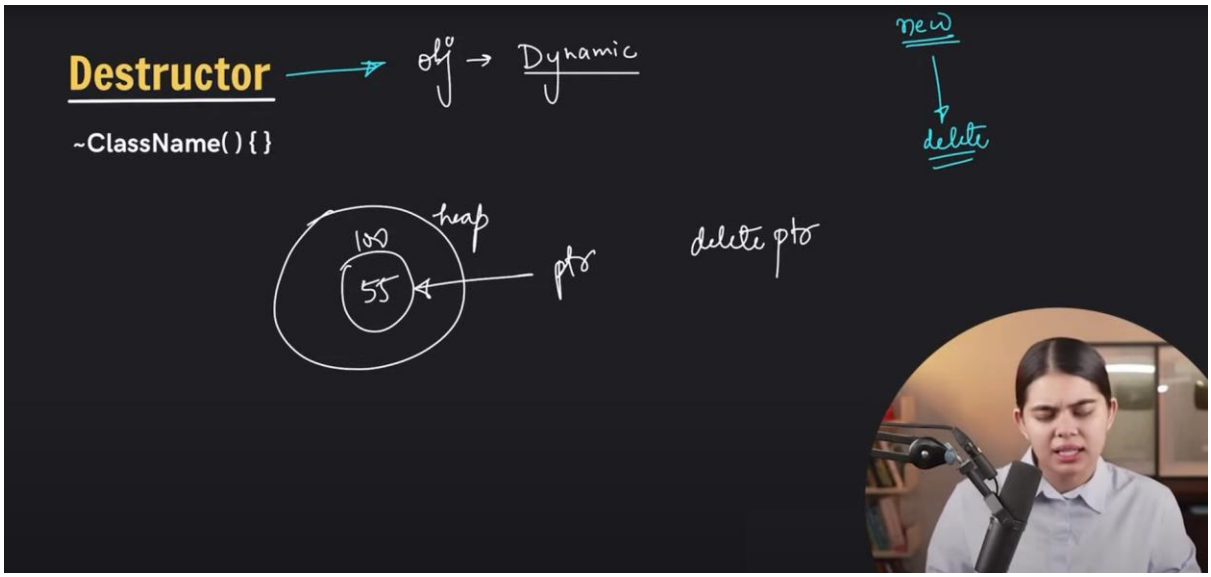
Destructor

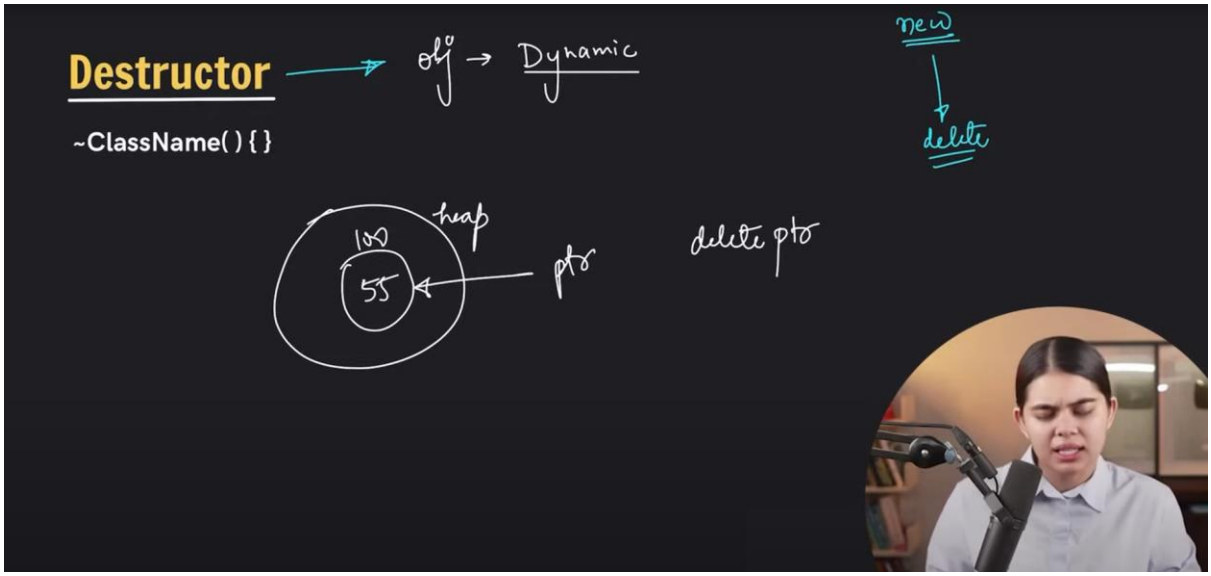
Compiler does this automatically

But agar dynamic memory allocation hai toh dynamic deallowcation karna padega

It does not delete pointer but delete memory to which the pointer points

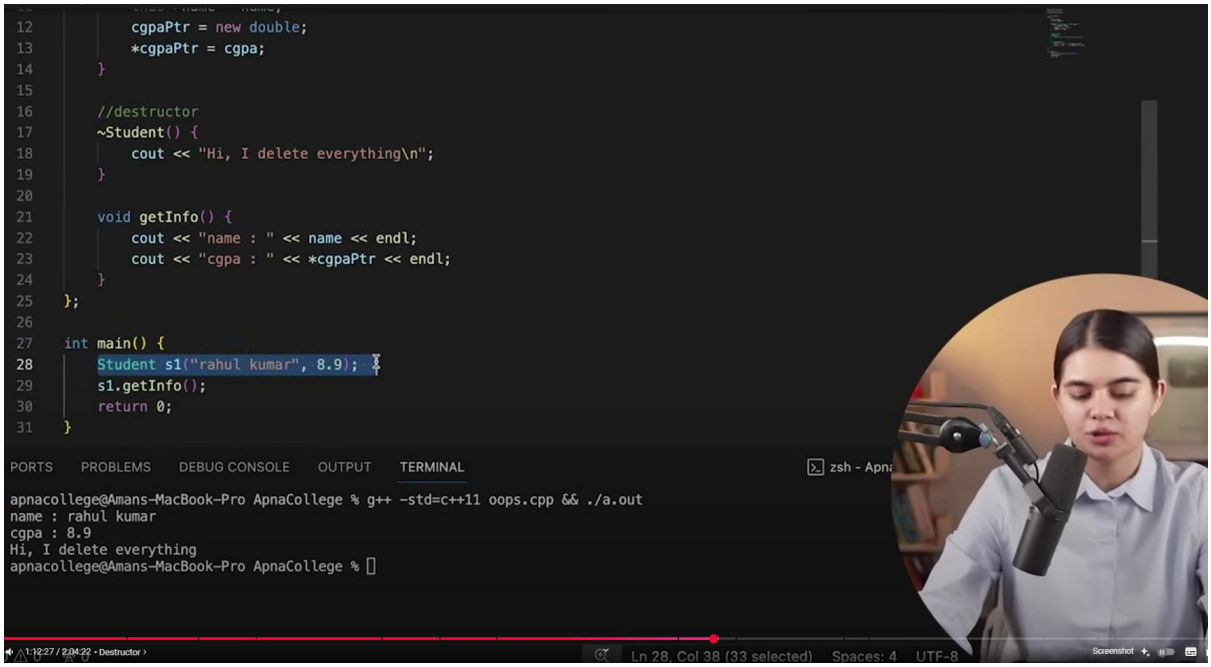
Cout<<ptr still works





Called automatically

~ starring with it

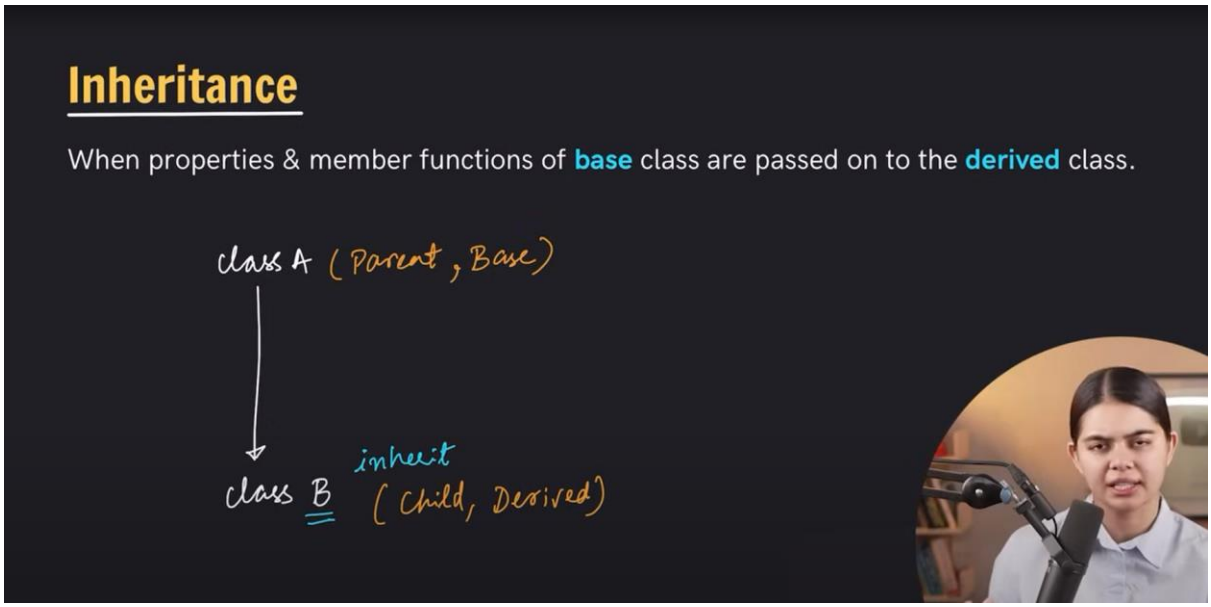


```
    //destructor
    ~Student() {
        cout << "Hi, I delete everything\n";
        delete cgpaPtr;
    }

    void getInfo() {
        cout << "name : " << name << endl;
        cout << "cgpa : " << *cgpaPtr << endl;
    }
};

int main() {
    Student s1("rahul kumar", 8.9);
    s1.getInfo();
    return 0;
}
```

If not done memory leak can happen

Inheritance

Use for code reuseability



In this we are doing in public mode of inheritance

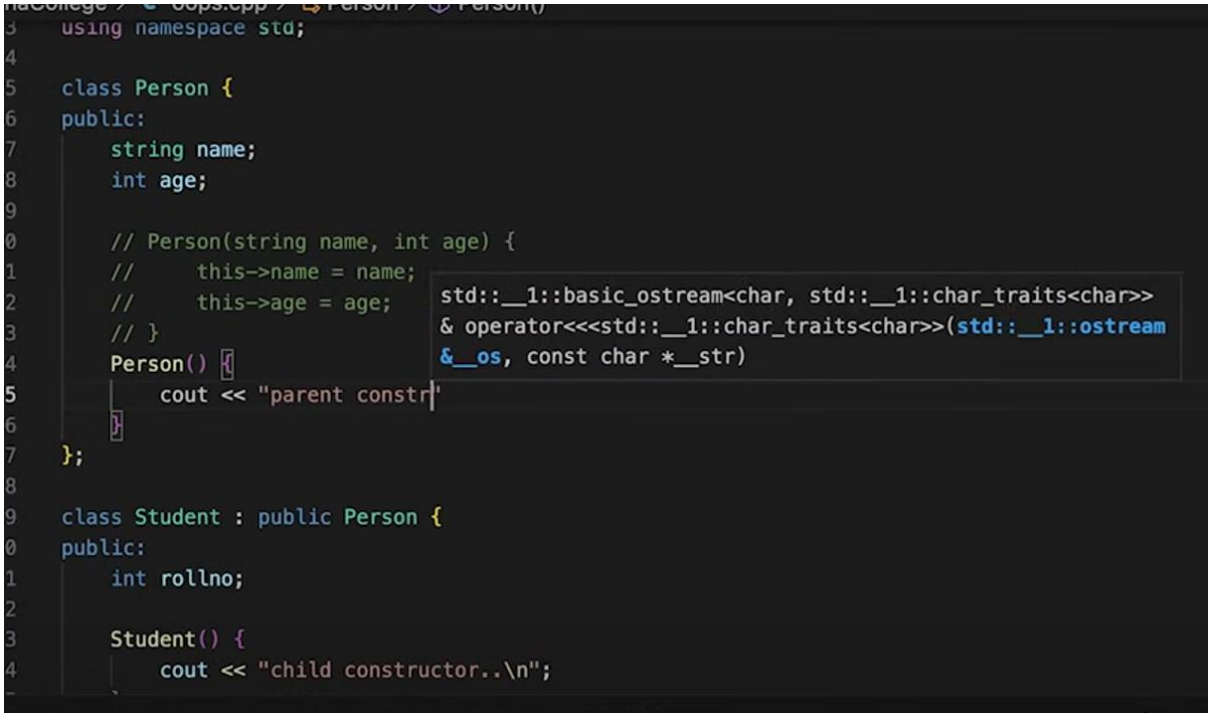
Pehle parent class ka constructor call hota fir child class ka

Opposite in destructor

```
class Person {
public:
    string name;
    int age;

    // Person(string name, int age) {
    //     this->name = name;
    //     this->age = age;
    // }
    Person() {

    }
};

class Student : public Person {
    //name, age, rollno
}
```

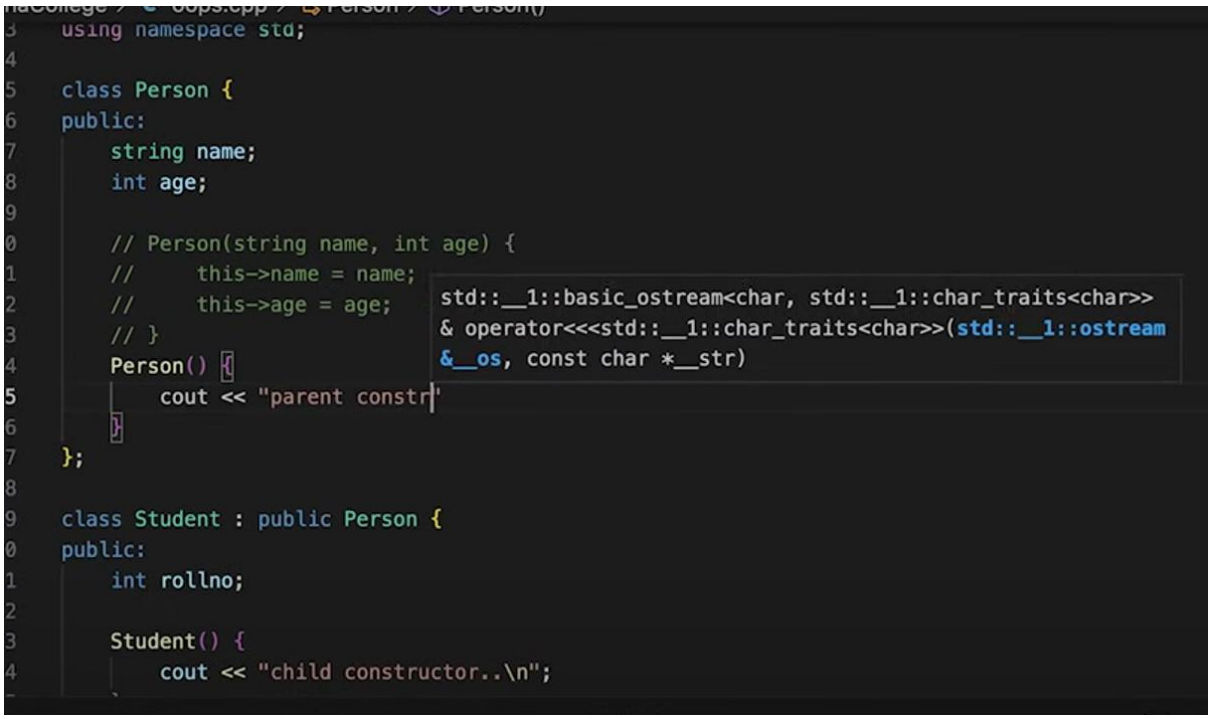
No properties made like name in student class but still we get

```
class Student : public Person {
public:
    int rollno;

    void getInfo() {
        cout << "name : " << name << endl;
        cout << "age : " << age << endl;
        cout << "rollno : " << rollno << endl;
    }
};

int main() {
    Student s1;
    s1.name = "rahul kumar";
    s1.age = 21;
    s1.rollno = 1234;

    s1.getInfo();
    return 0;
```



```
using namespace std;

class Person {
public:
    string name;
    int age;

    // Person(string name, int age) {
    //     this->name = name;
    //     this->age = age;
    // }
    Person() {
        cout << "parent constr"
    }
};

class Student : public Person {
public:
    int rollno;

    Student() {
        cout << "child constructor..\n";
```

```
std::__1::basic_ostream<char, std::__1::char_traits<char>>
& operator<<<std::__1::char_traits<char>>(std::__1::ostream
&__os, const char *__str)
```



```
using namespace std;

class Person {
public:
    string name;
    int age;

    // Person(string name, int age) {
    //     this->name = name;
    //     this->age = age;
    // }
    Person() {
        cout << "parent constr"
    }
};

class Student : public Person {
public:
    int rollno;

    Student() {
        cout << "child constructor..\n";
```

```
std::__1::basic_ostream<char, std::__1::char_traits<char>>
& operator<<<std::__1::char_traits<char>>(std::__1::ostream
&__os, const char *__str)
```

Agar parameterised constructor hai parent me toh use call karenge aise

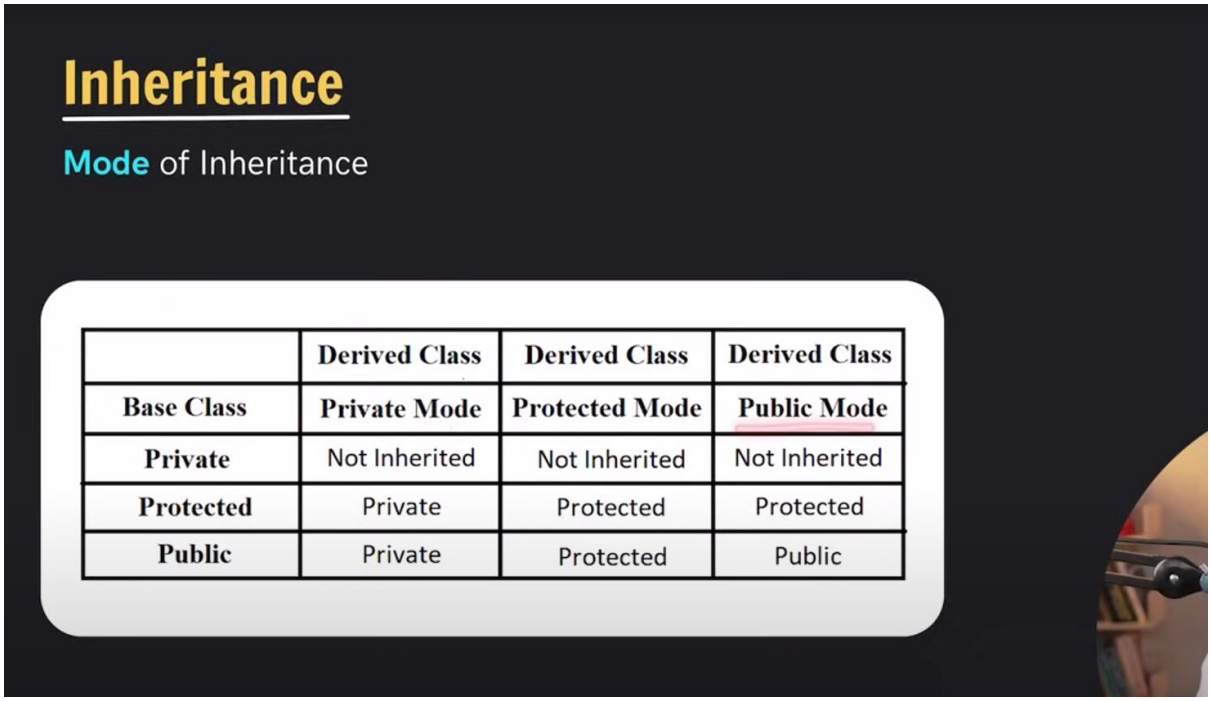
Parent constructor me type bhejne ki need nahi hai

```
13      }
14  };
15
16  class Student : public Person {
17  public:
18      int rollno;
19
20      Student(string name, int age, int rollno) : Person(name, age){
21          this->rollno = rollno;
22      }
23      void getInfo() {
24          cout << "name : " << name << endl;
25          cout << "age : " << age << endl;
26          cout << "rollno : " << rollno << endl;
27      }
28  };
29
30  int main() {
31      Student s1("rahul kumar", 21, 1234);
```

PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

Ye abtak public mode of inheritance thi

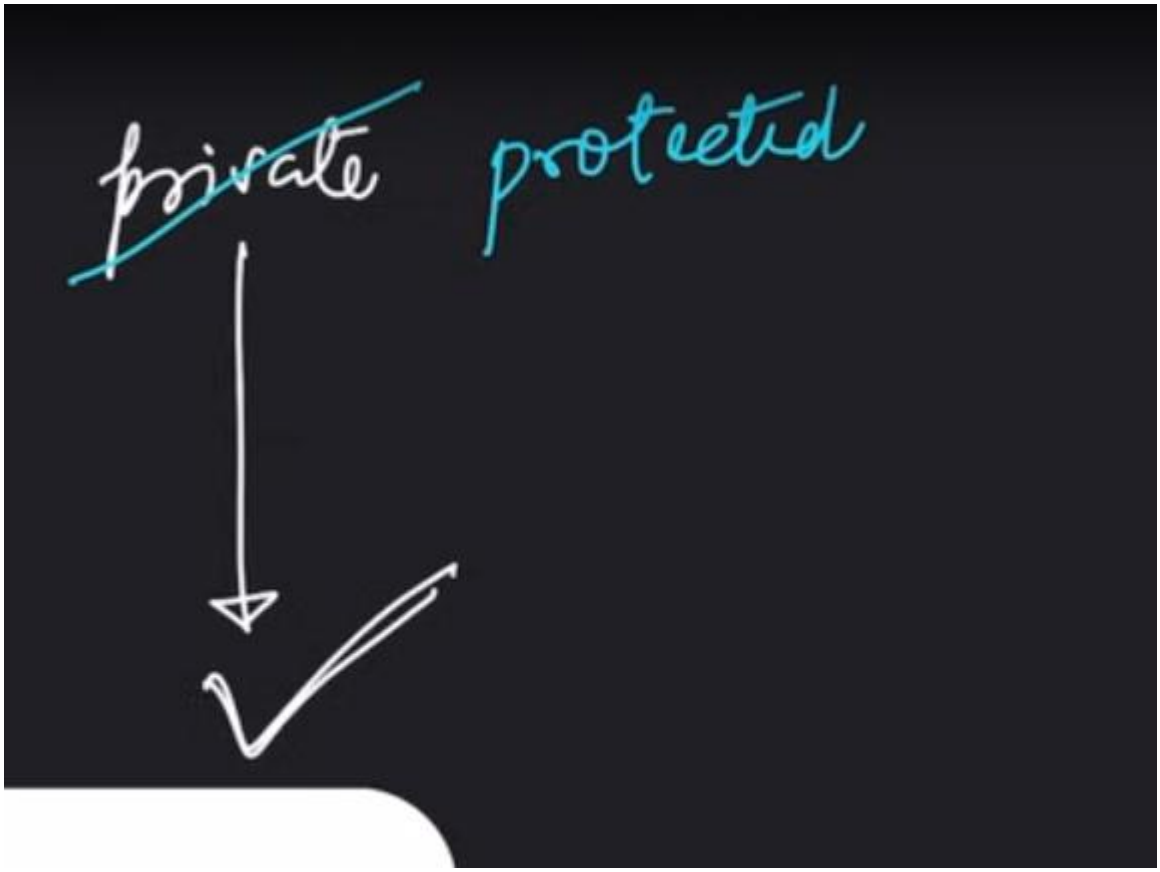
Modes of inhertitance



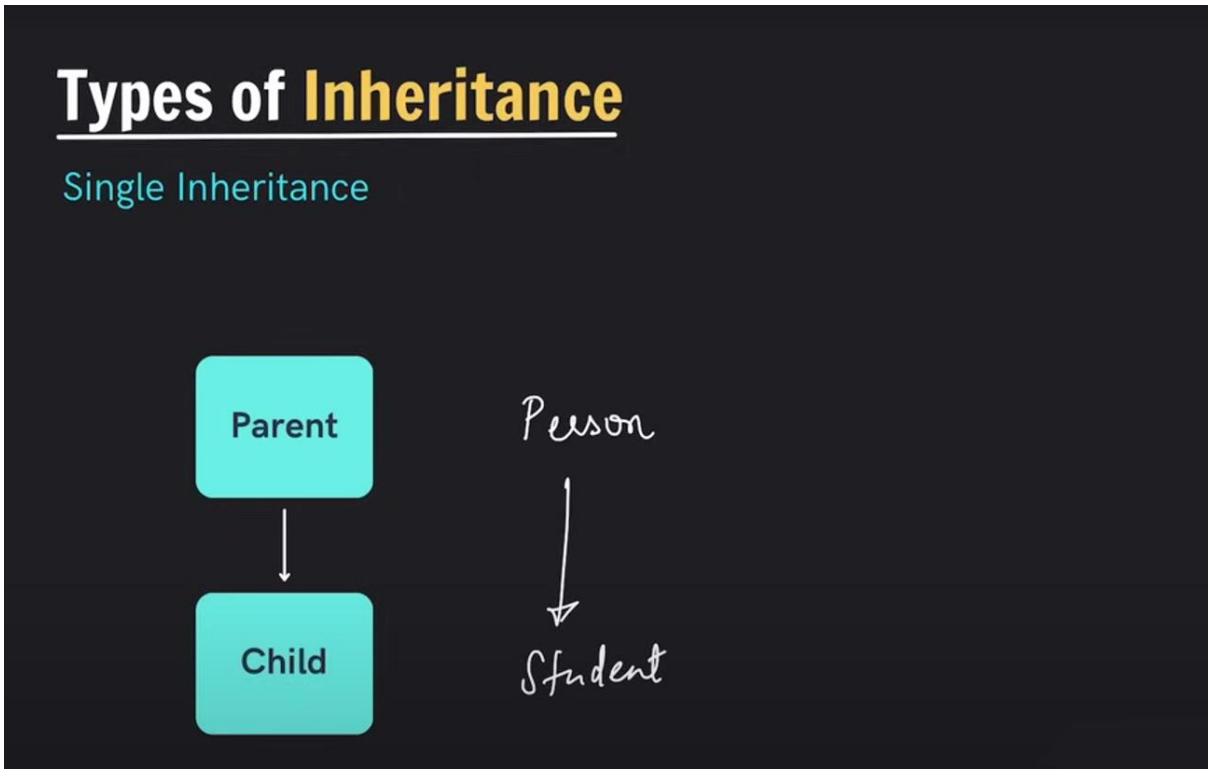
	Derived Class	Derived Class	Derived Class
Base Class	Private Mode	Protected Mode	Public Mode
Private	Not Inherited	Not Inherited	Not Inherited
Protected	Private	Protected	Protected
Public	Private	Protected	Public

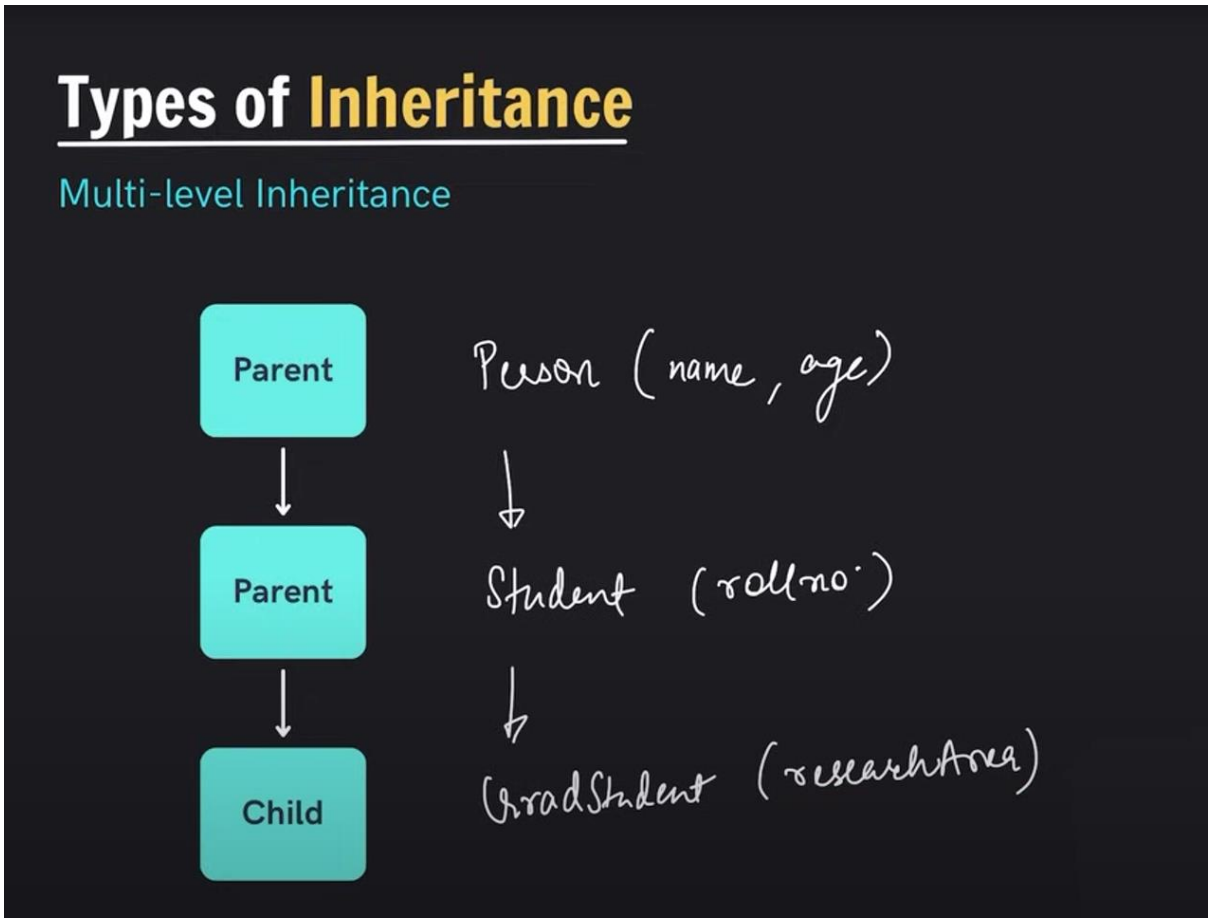
If we have some private property that we just want to use for inheritance and not give them access anywhere else we make them protected

Sirf inhetitance ke liye pass karna inhe



Types of inhetitance

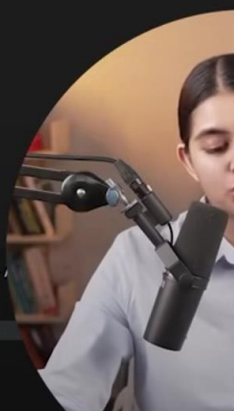




```cpp
#include <string>
using namespace std;

class Person {
public:
    string name;
    int age;
};

class Student : public Person {
public:
    int rollno;
};

class GradStudent : public Student {
public:
    string researchArea;
};

int main() {
    GradStudent s1;
    s1.name = "tony stark";
    s1.researchArea = "quantum physics";
```

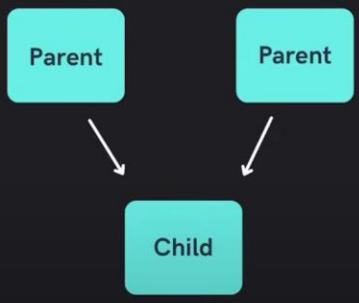
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL zsh -

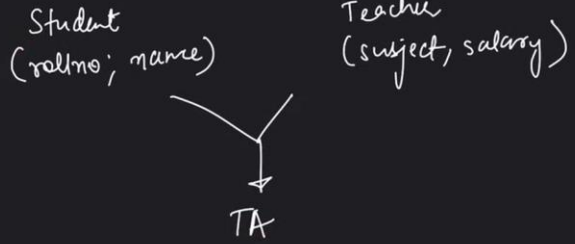
```
nacollege@Amans-MacBook-Pro ApnaCollege % g++ -std=c++11 oops.cpp && ./a.out
ny stark
uantum physics
nacollege@Amans-MacBook-Pro ApnaCollege %
```

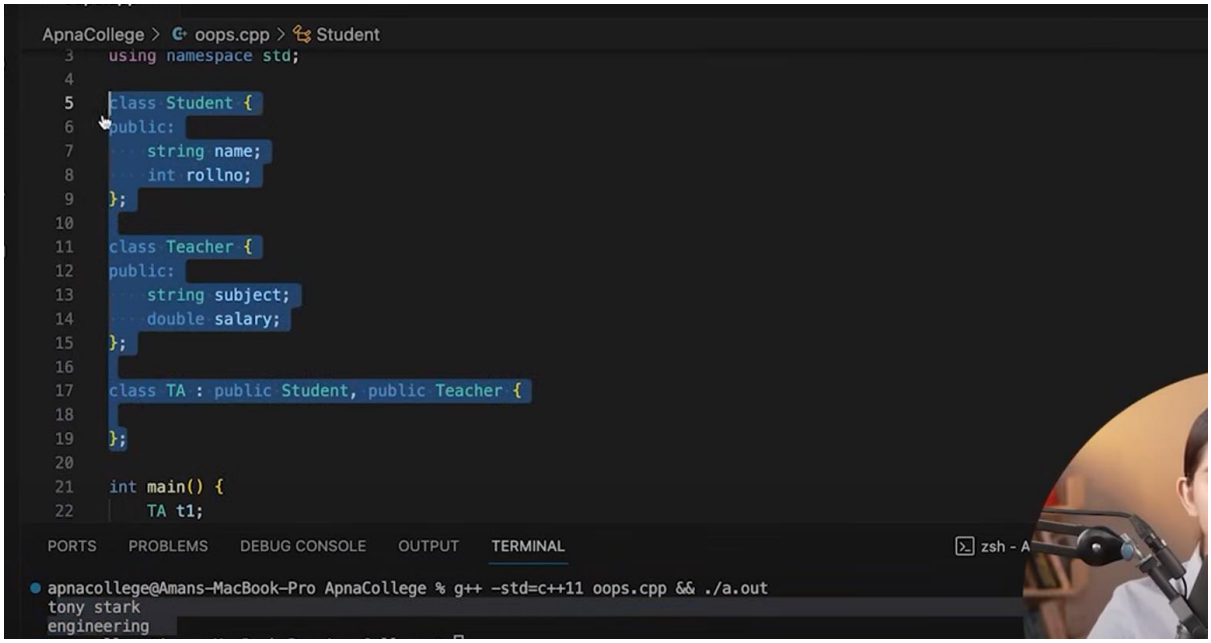


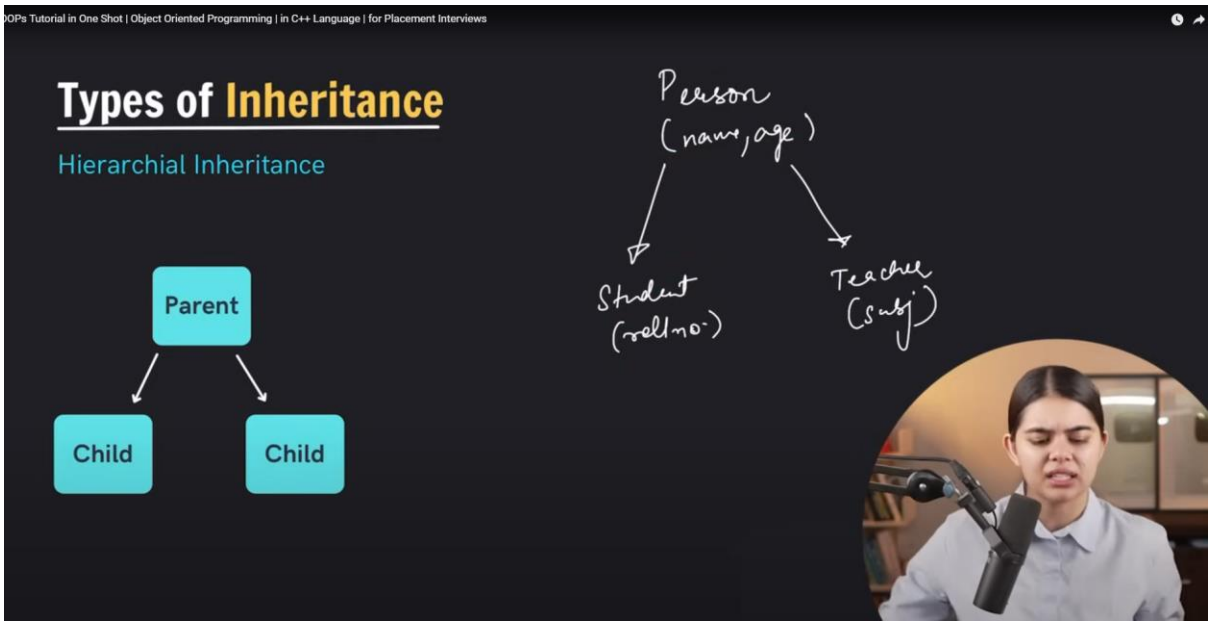
Types of Inheritance

Multiple Inheritance



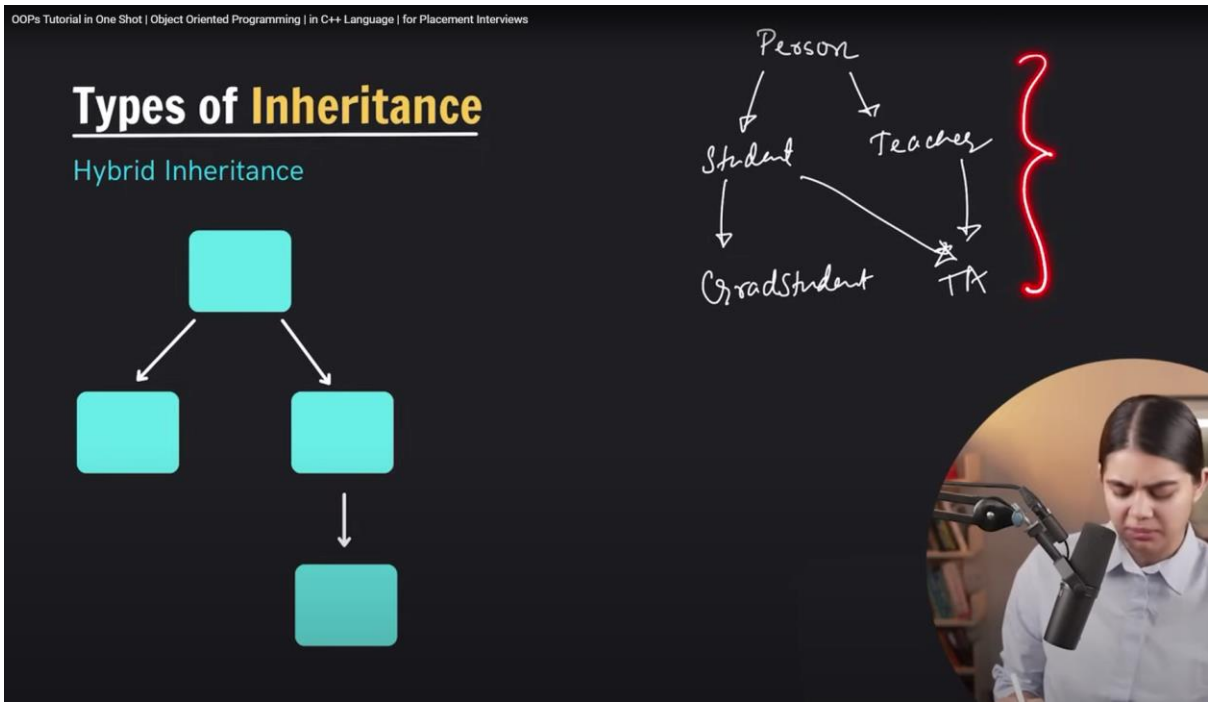




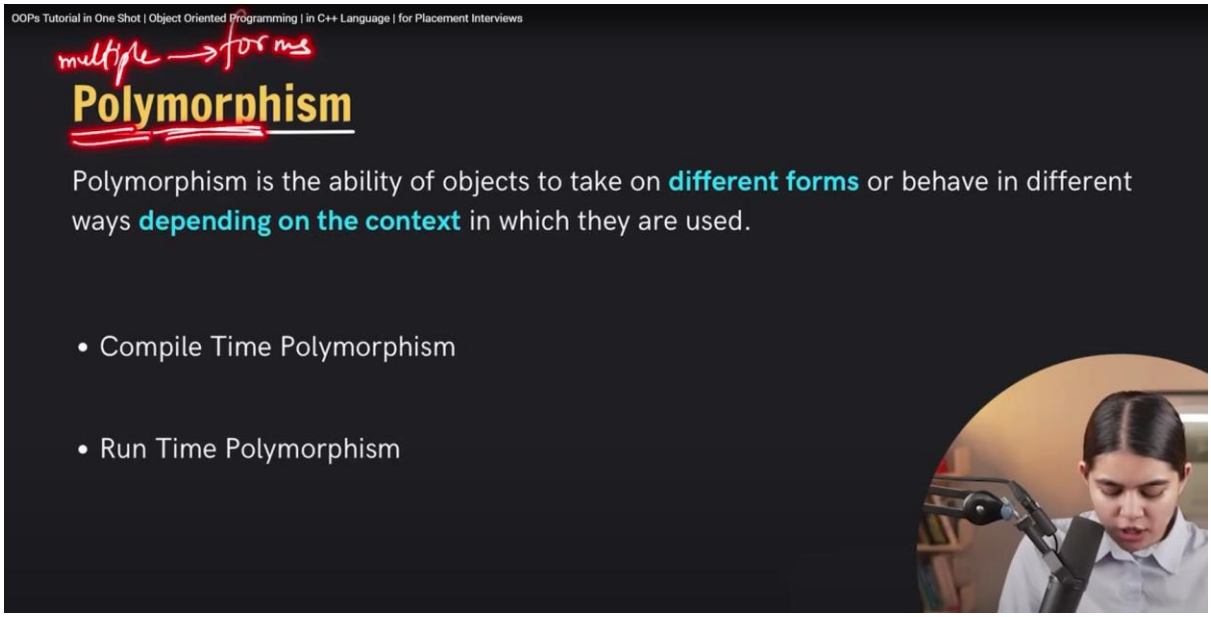


For consistacy classes ka name capital me hi Rakha jata hai

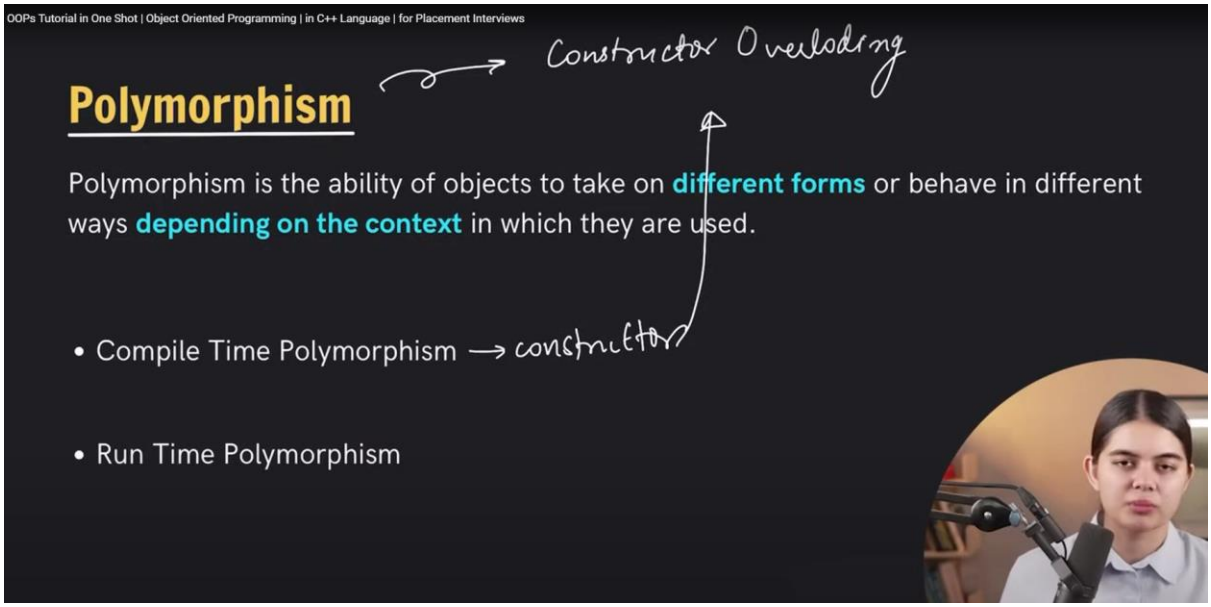
```cpp
class Person {
public:
    string name;
    int age;
};

class Student : public Person {
public:
    int rollno;
};

class Teacher : public Person{
public:
    string subject;
};

class TA : public Student, public Teacher {
```



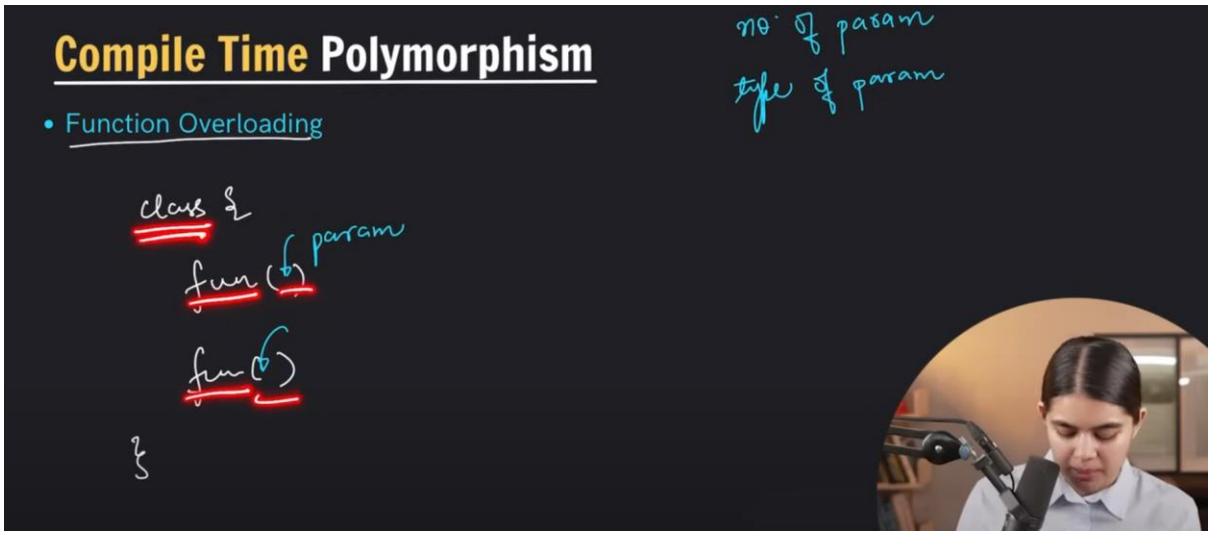
Polymorphism



Constructor overloading is example of polymorphism (compile time)

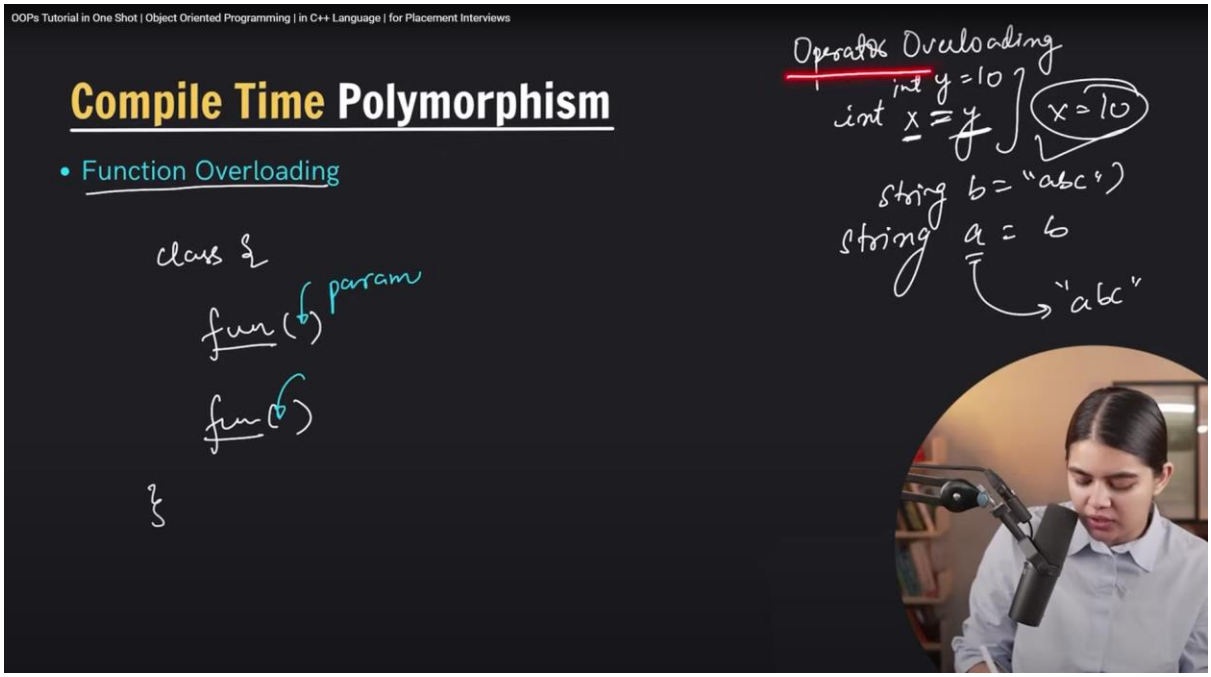


Function overloading is implementation of compile time polymorphism and it is when we are defining 2 or more functions with same name in same class but they only differ in terms of their parameters (no. or type of parameters)




```cpp
using namespace std;

class Print {
public:
    void show(int x) {
        cout << "int : " << x << endl;
    }

    void show(char ch) {
        cout << "char : " << ch << endl;
    }
};

int main() {
    Print p1;
    p1.show('&');
    return 0;
}
```

```
PORTS  PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL

apnacollege@Amans-MacBook-Pro ApnaCollege % g++ -std=c++11 oops.cpp && ./a.out
int : 101
apnacollege@Amans-MacBook-Pro ApnaCollege % g++ -std=c++11 oops.cpp && ./a.out
char : &
apnacollege@Amans-MacBook-Pro ApnaCollege %
```



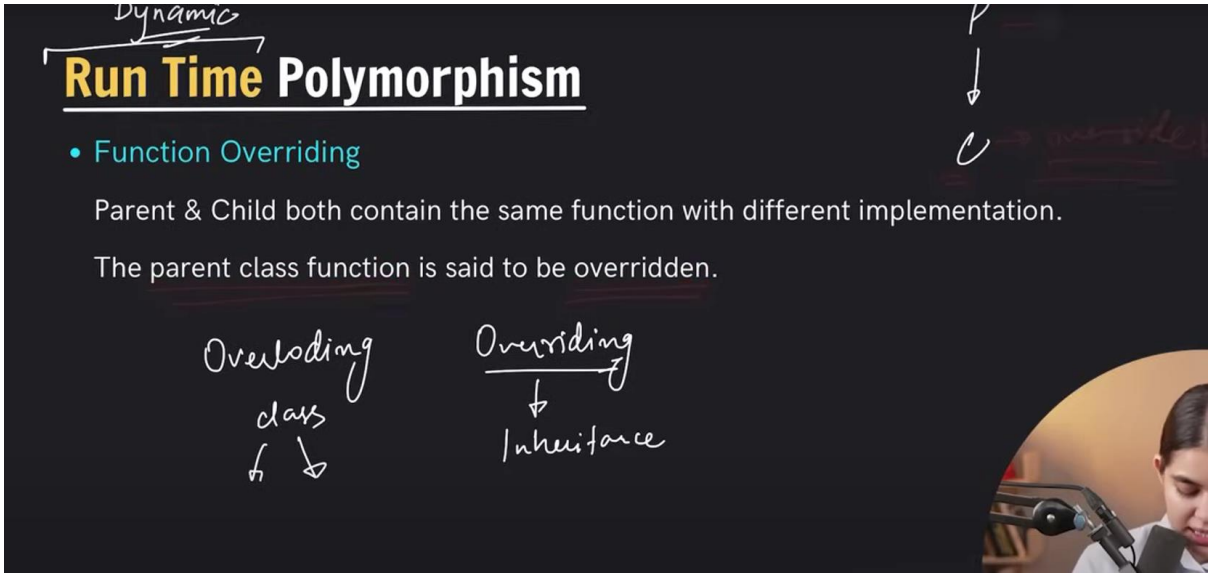
Compile time polymorphism ex

Constructor overloading

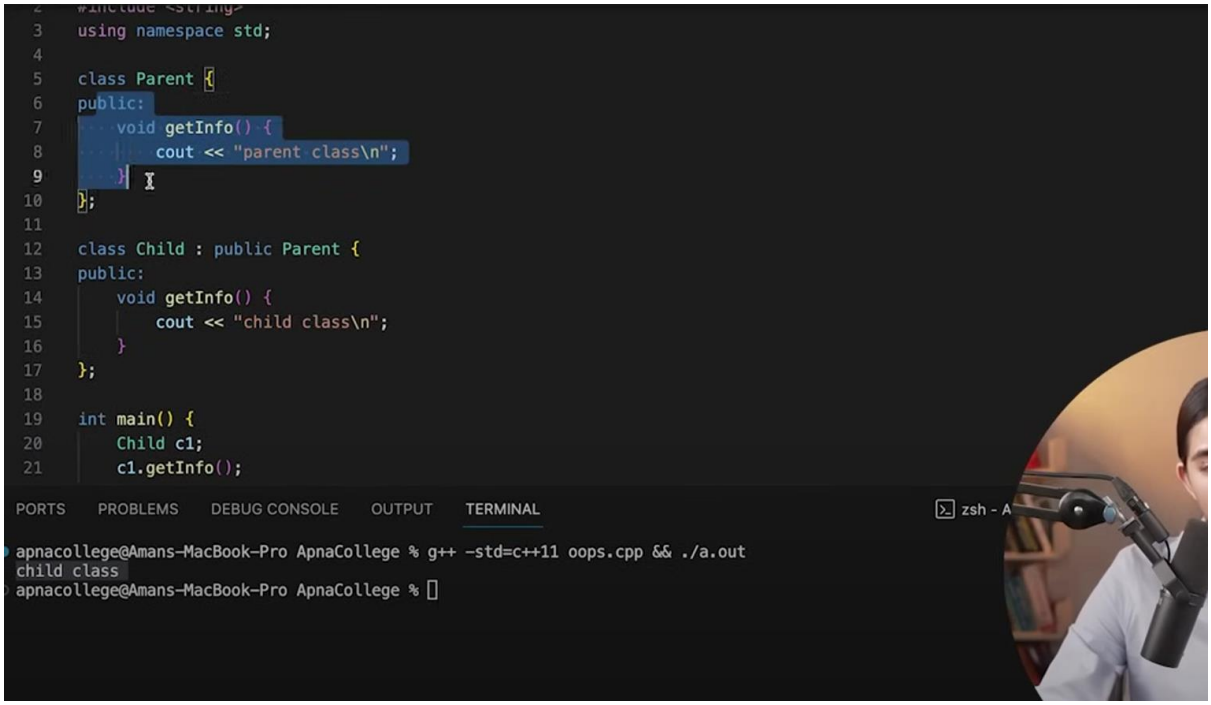
Function overloading

Operator overloading

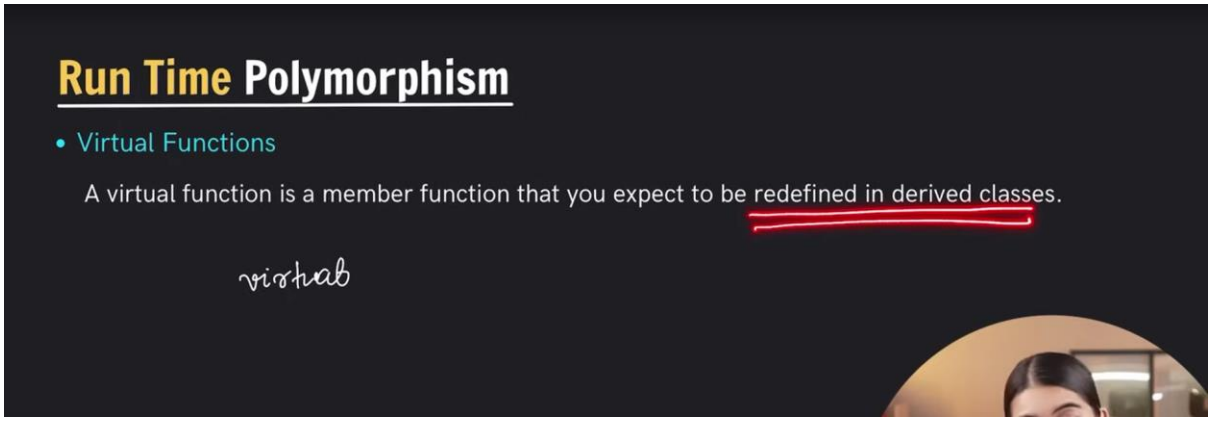
Overloading me within class 2 function



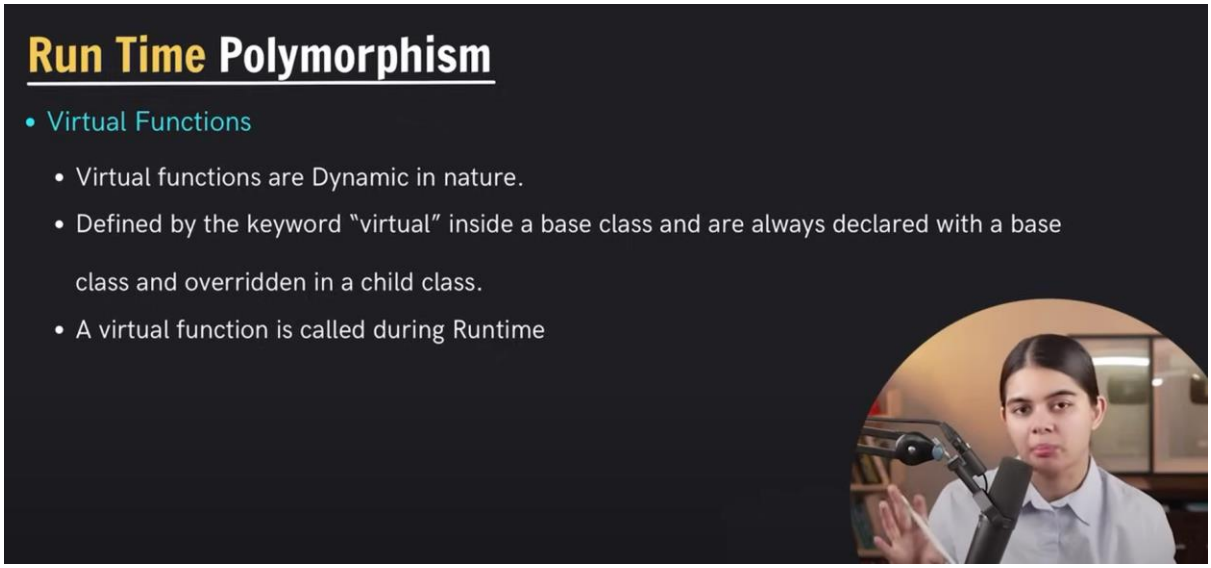
If object is of child class child function is called if parent class obj then parent class function caleed

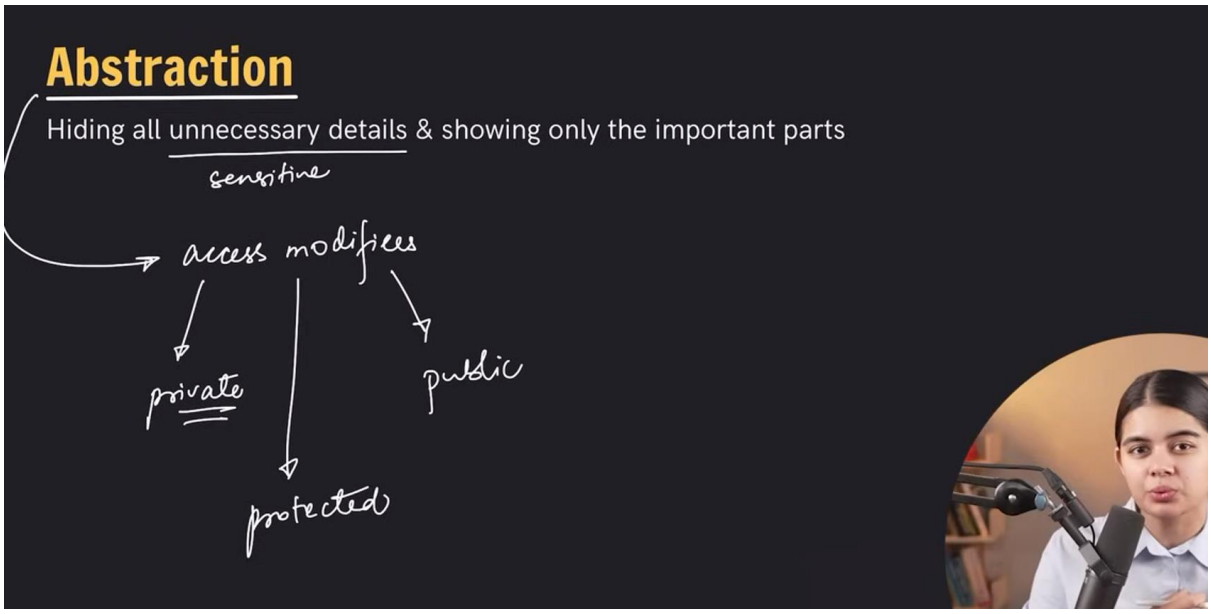


Child class function override parent class

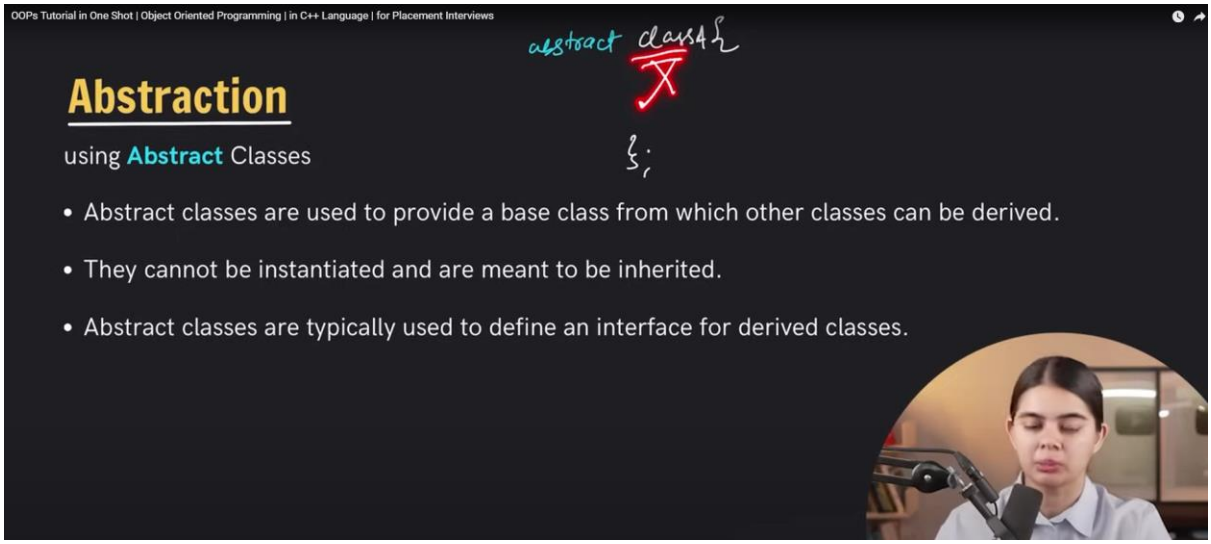


Almost like overriding





Data hiding only hides but abstraction also shows useful info



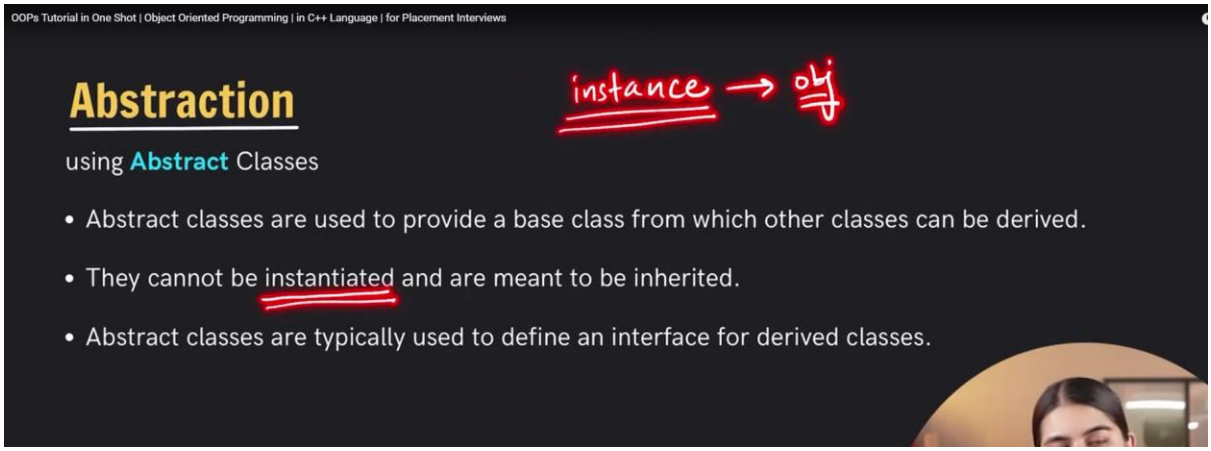
Ye class ka object nahi banta ye dusri class ka blueprint banti hai

Made only for inheritance

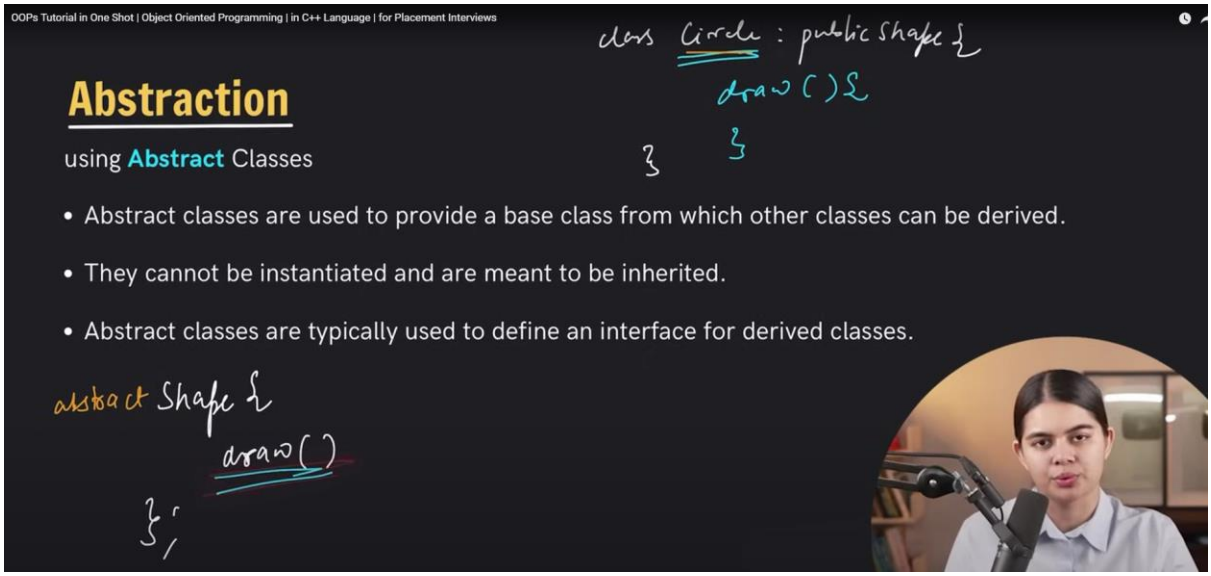
NOT POSSIBLE TO MAKE OBJECTS

Abstact class ka kaam hai bas ek idea dena

Ye sirf ek blue print deti hai konsi chije base class me honi chaiye

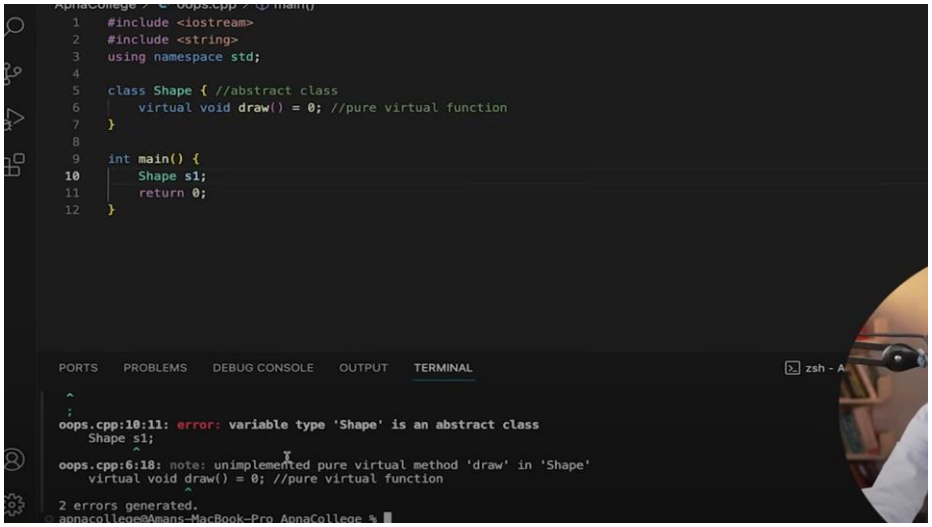


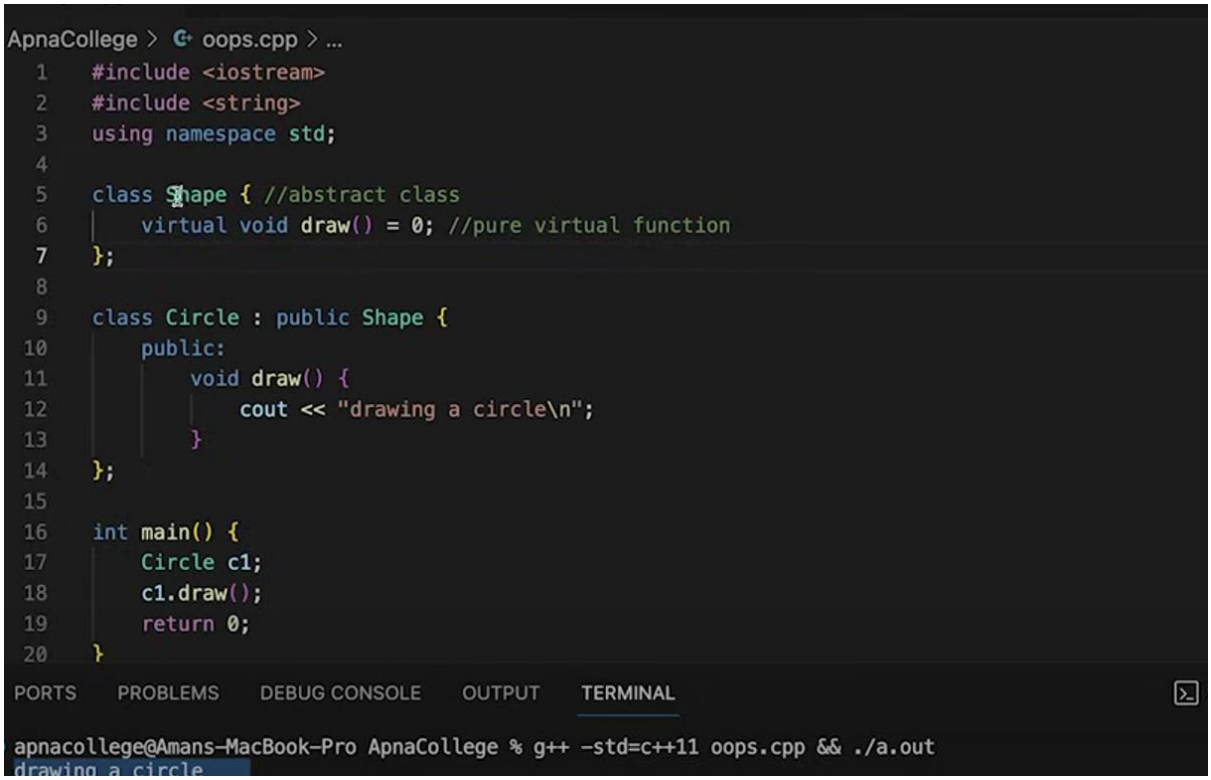
Make any shape in this example but they should have draw function



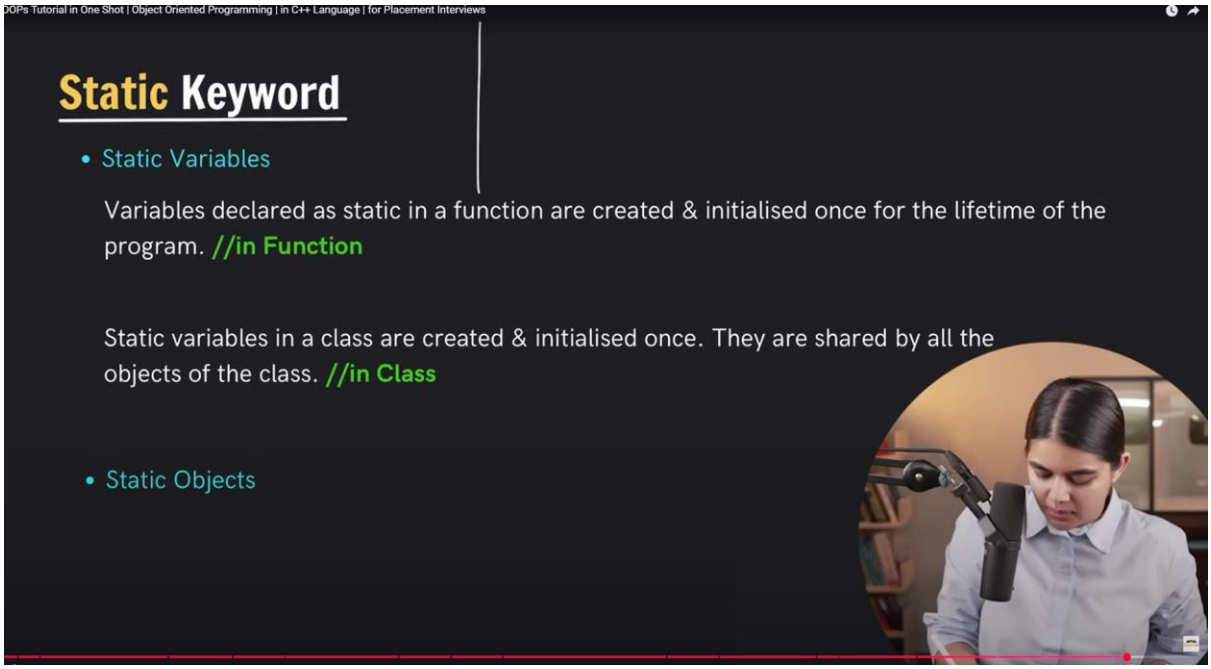
Jis class me pure virtual function hota hai who automatically abstract class ban jati hai

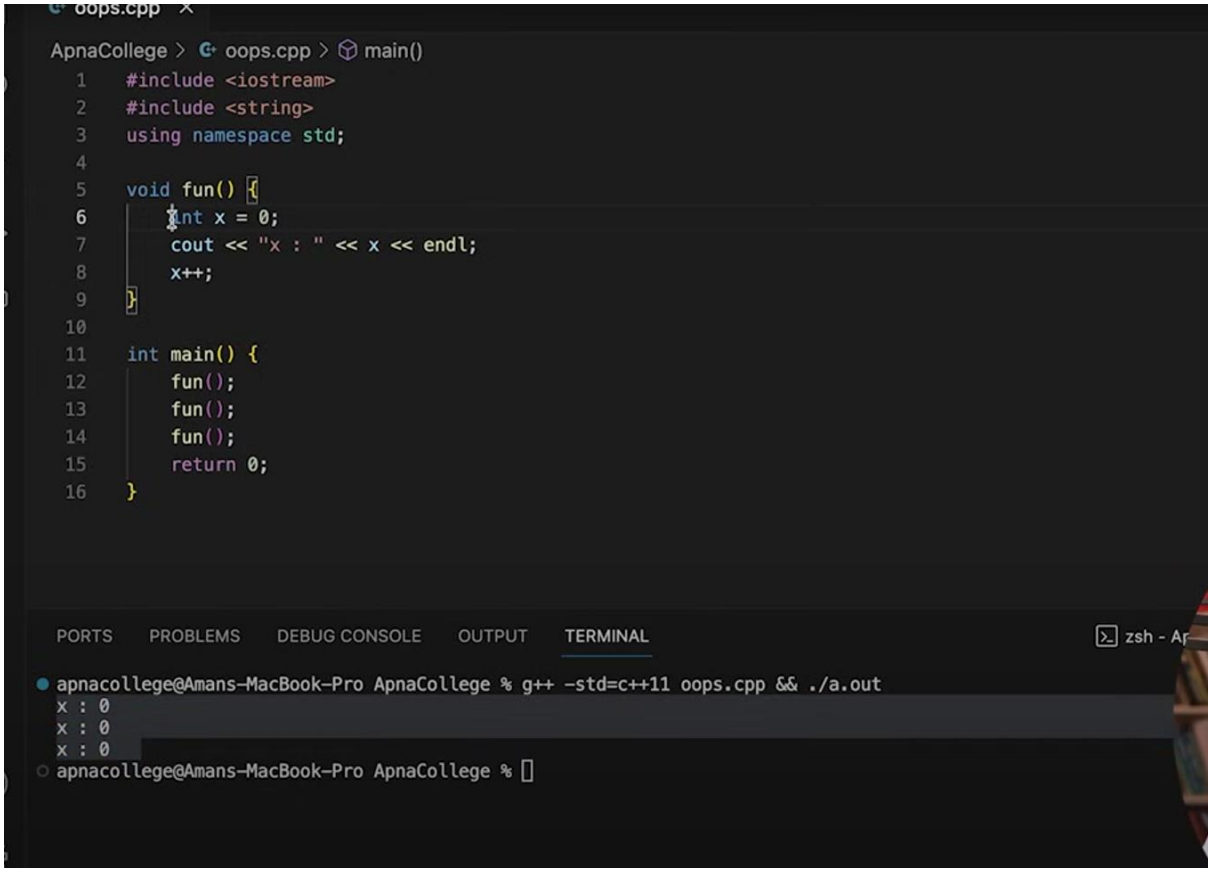
```
class Shape {
    virtual void draw() = 0; //pure virtual function
};

int main() {

    return 0;
}
```


#include <iostream>
#include <string>
using namespace std;
class Shape { //abstract class
virtual void draw() = 0; //pure virtual function
}
int main() {
Shape s1;
return 0;
}
PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
oops.cpp:10:11: error: variable type 'Shape' is an abstract class
Shape s1;
oops.cpp:6:18: note: unimplemented pure virtual method 'draw' in 'Shape'
virtual void draw() = 0; //pure virtual function
2 errors generated.


ApnaCollege > oops.cpp > ...
#include <iostream>
#include <string>
using namespace std;
class Shape { //abstract class
virtual void draw() = 0; //pure virtual function
};
class Circle : public Shape {
public:
void draw() {
cout << "drawing a circle\n";
}
};
int main() {
Circle c1;
c1.draw();
return 0;
}
PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
apnacollege@Amans-MacBook-Pro ApnaCollege % g++ -std=c++11 oops.cpp && ./a.out
drawing a circle

Static


OOPs Tutorial in One Shot | Object Oriented Programming | in C++ Language | for Placement Interviews
Static Keyword
• Static Variables
Variables declared as static in a function are created & initialised once for the lifetime of the program. //in Function
Static variables in a class are created & initialised once. They are shared by all the objects of the class. //in Class
• Static Objects


oops.cpp
ApnaCollege > oops.cpp > main()
#include <iostream>
#include <string>
using namespace std;
void fun() {
int x = 0;
cout << "x : " << x << endl;
x++;
}
int main() {
fun();
fun();
fun();
return 0;
}
PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
zsh - Ap
apnacollege@Amans-MacBook-Pro ApnaCollege % g++ -std=c++11 oops.cpp && ./a.out
x : 0
x : 0
x : 0
apnacollege@Amans-MacBook-Pro ApnaCollege %

Static Keyword

- Static Variables

Variables declared as static in a function are created & initialised once for the lifetime of the program. //in Function

OOPs Tutorial in One Shot | Object Oriented Programming | in C++ Language | for Placement Interviews

oops.cpp

ApnaCollege > oops.cpp > fun()

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

void fun() {
    static int x = 0; //init statement - 1 run
    cout << "x : " << x << endl;
    x++;
}

int main() {
    fun();
    fun();
    fun();
    return 0;
}
```

PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

```
apnacollege@Amans-MacBook-Pro ApnaCollege % g++ -std=c++11 oops.cpp && ./a.out
x : 0
x : 1
x : 2
apnacollege@Amans-MacBook-Pro ApnaCollege %
```

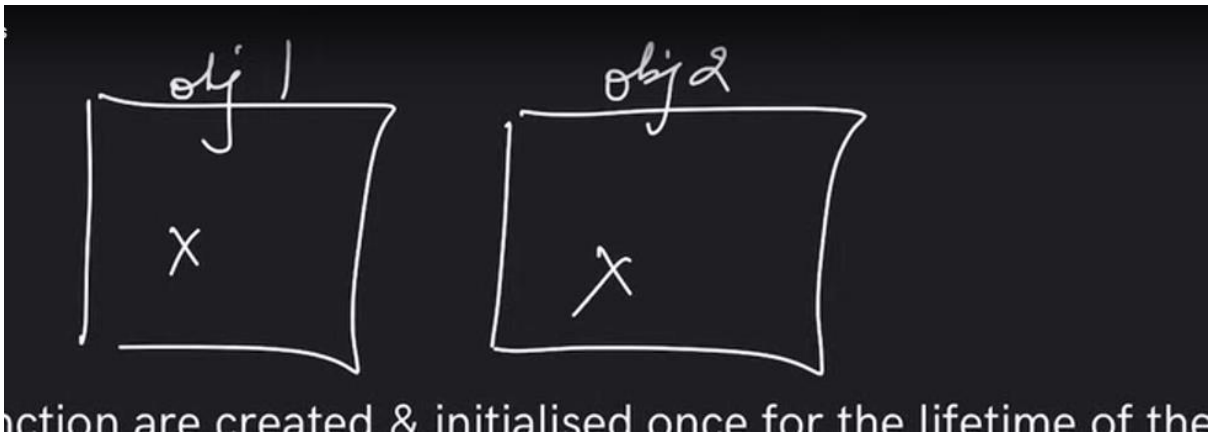
```
class A {
public:
    int x;

    void incX() {
        x = x + 1;
    }
};
```

Normal without static in class

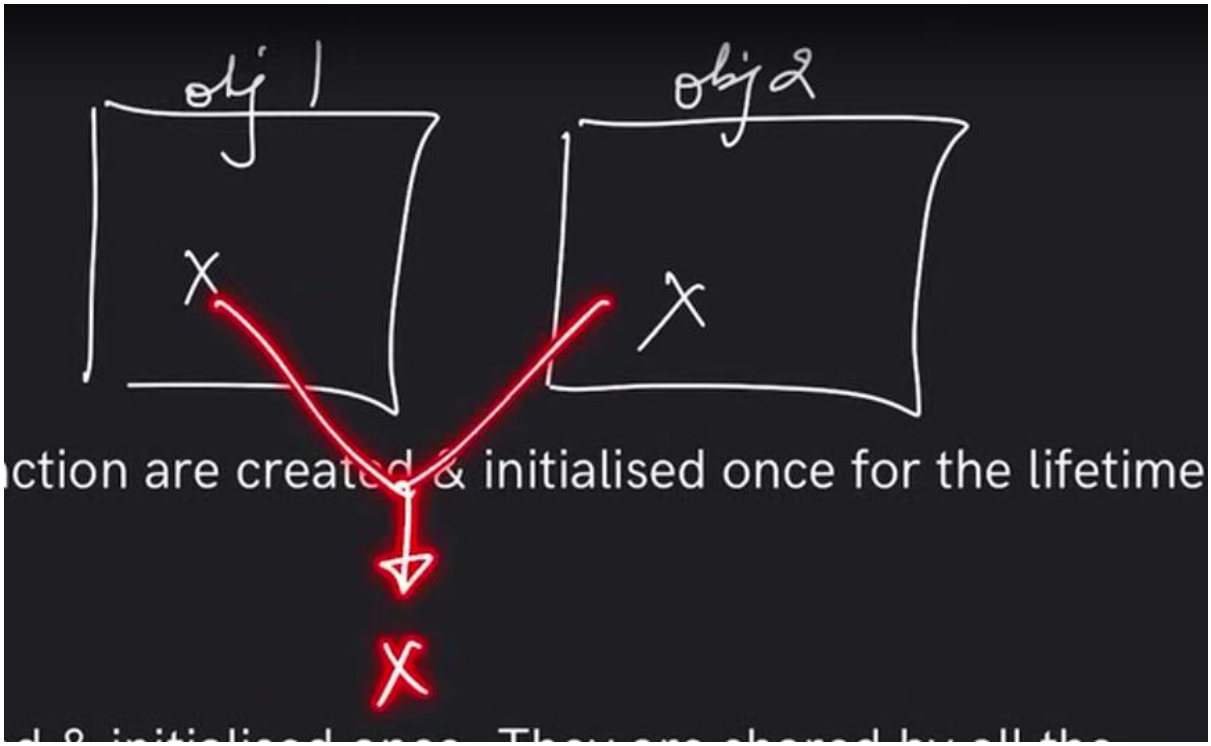
```
    }
};

int main() {
    A obj1;
    A obj2;

    obj1.x = 100;
    obj2.x = 200;

    cout << obj1.x << endl;

    return 0;
}
```

```
PORTS  PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL

apnacollege@Amans-MacBook-Pro ApnaCollege % g++ -std=c++11 oops.cpp && ./a.out
200
apnacollege@Amans-MacBook-Pro ApnaCollege % g++ -std=c++11 oops.cpp && ./a.out
100
apnacollege@Amans-MacBook-Pro ApnaCollege %
```

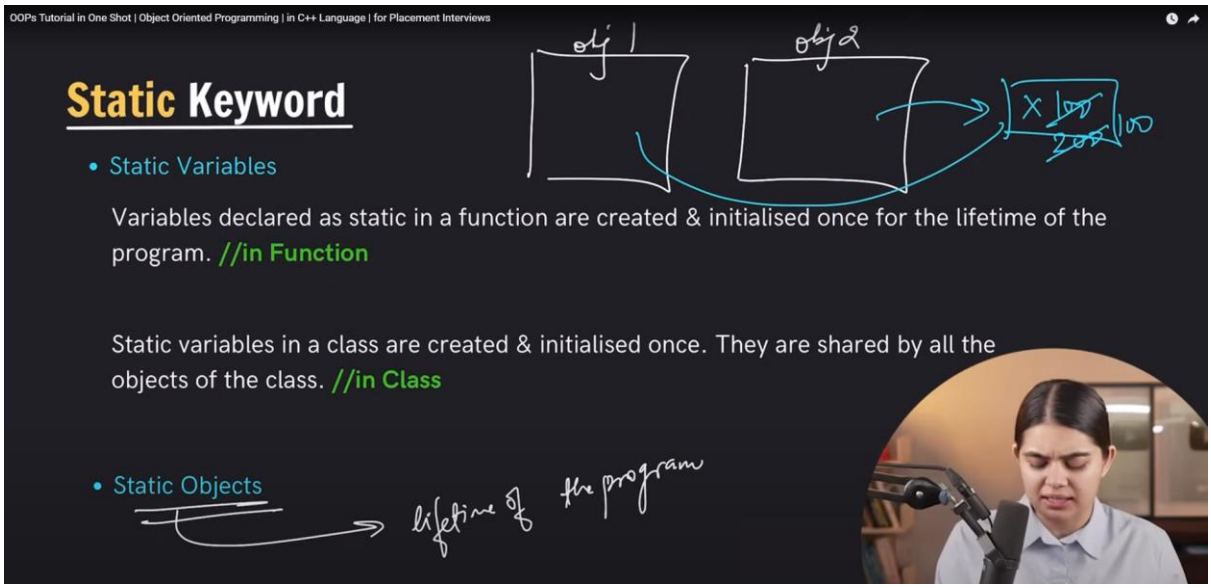
Without static



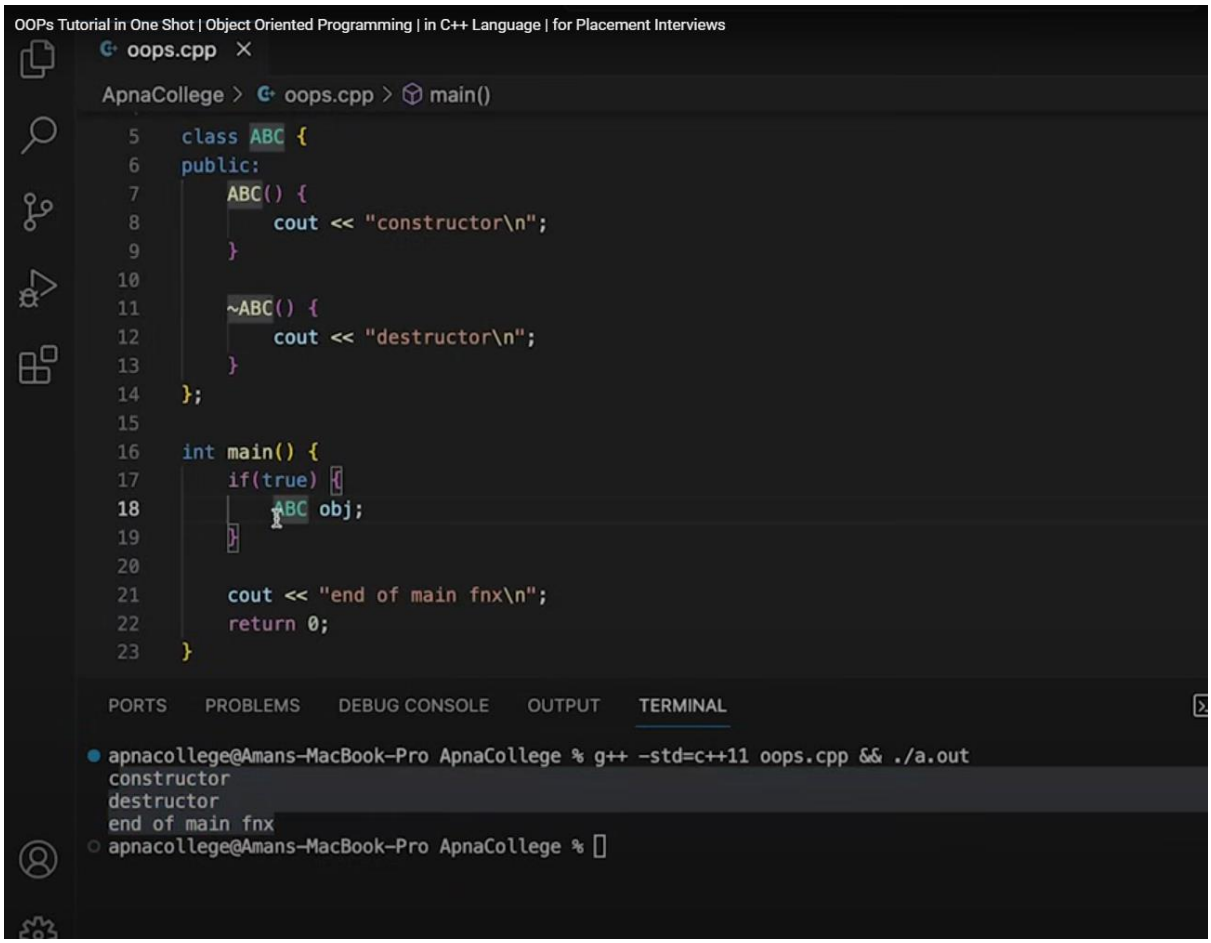
With static



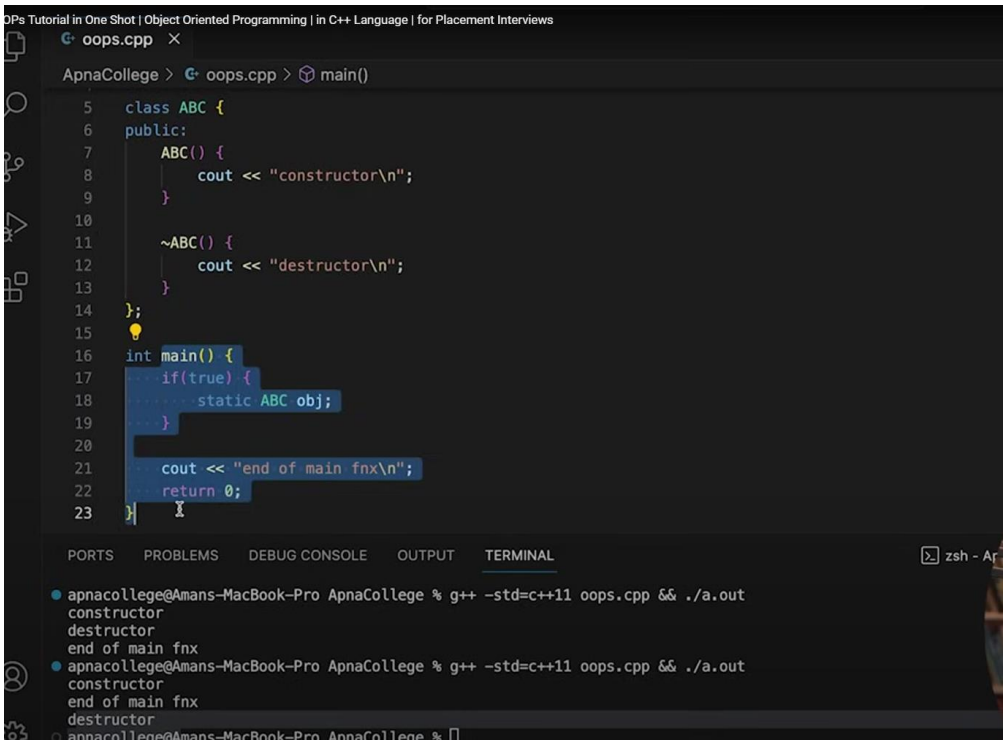
Static objects remain till the lifetime of program



Without static

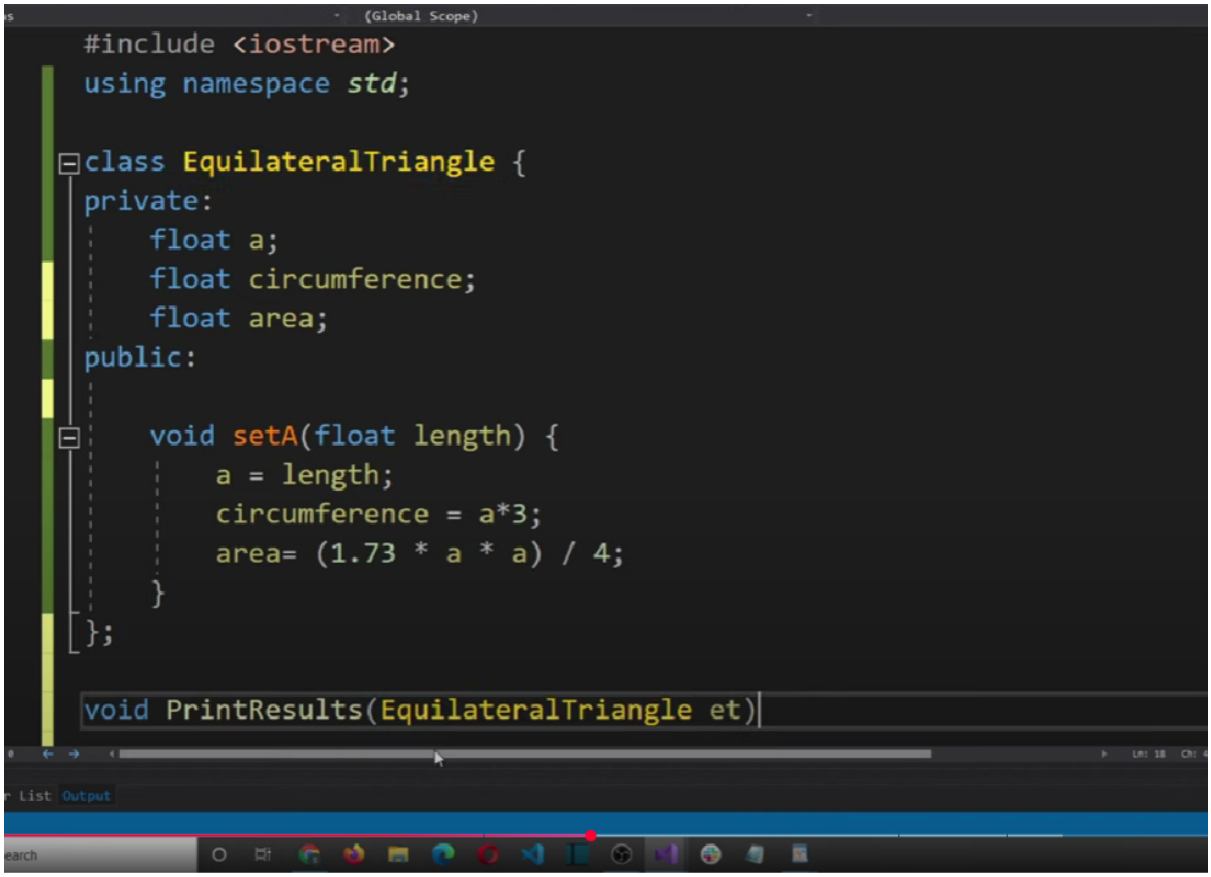


With static



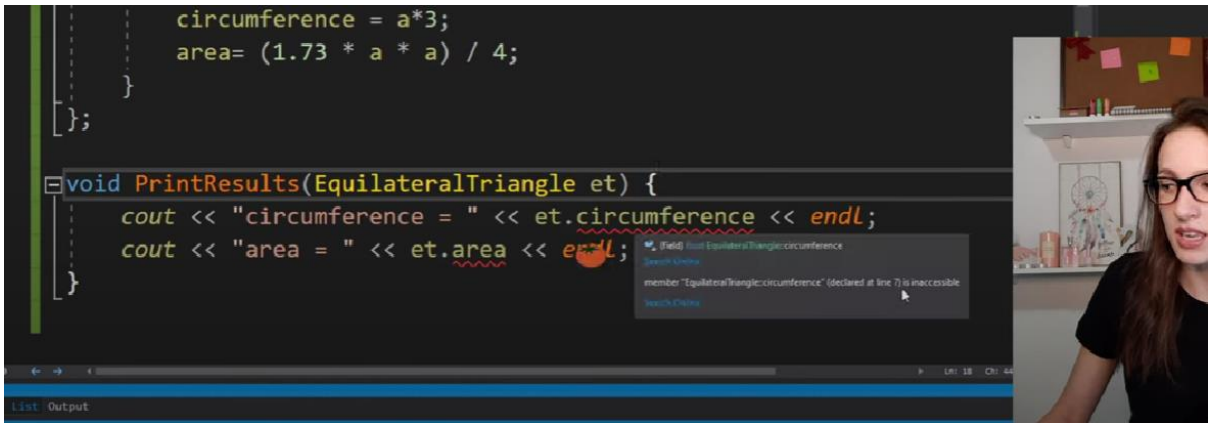
Friend function and friend class

Similar to real world friend will have access to private properties of class

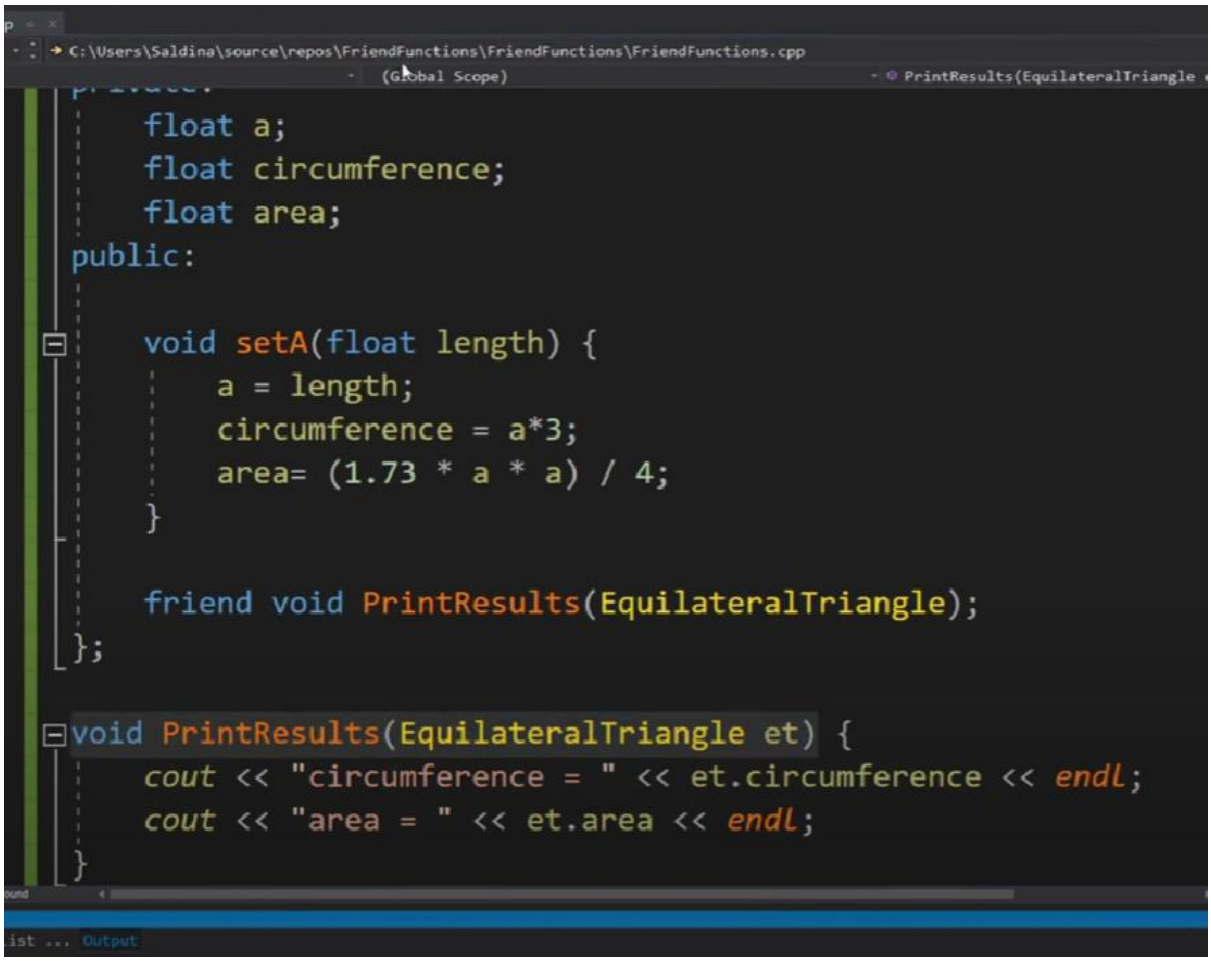


Other function

With



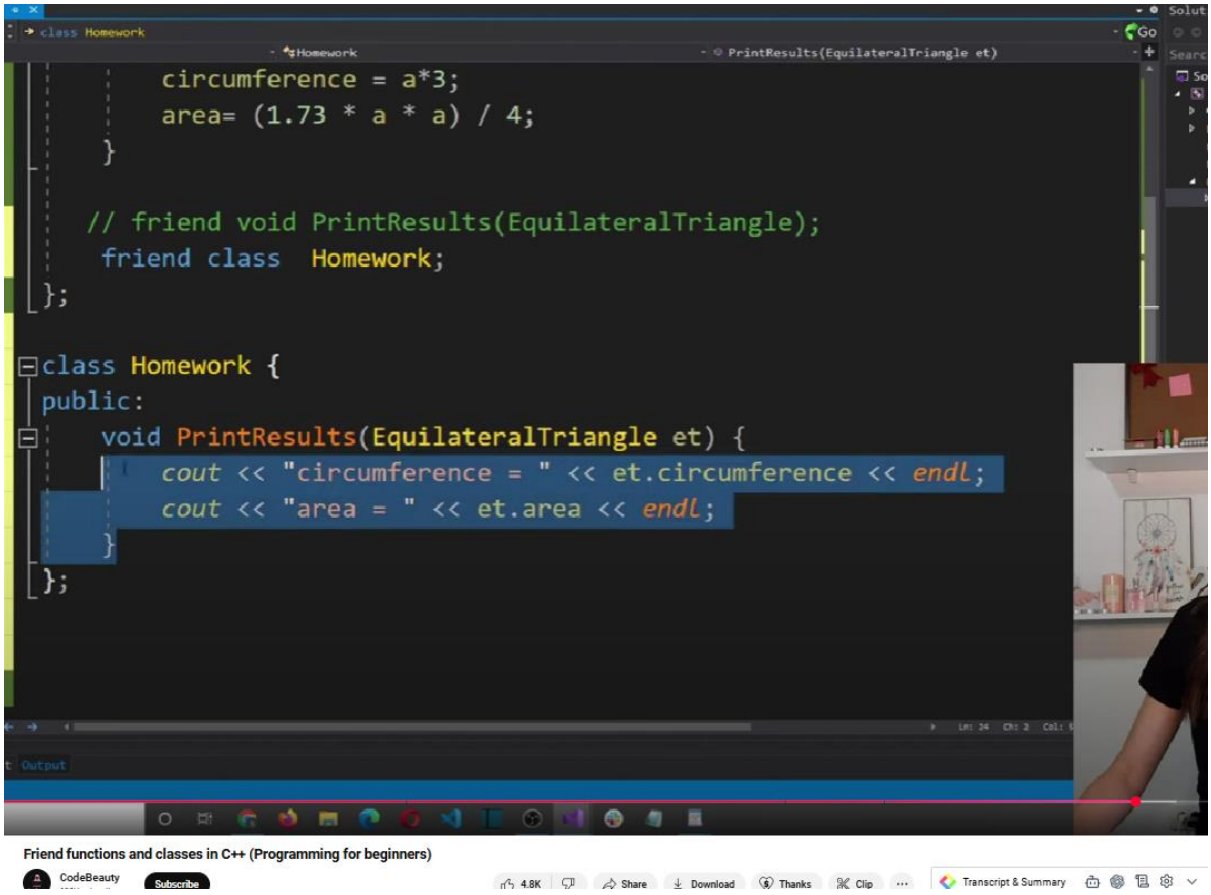
Write fried and just copy invocation of that



We have access to private properties called circumference and area in other func called printresult because we have made that function friend of our class

We don’t have access to these private properties outside friend function

Friend class

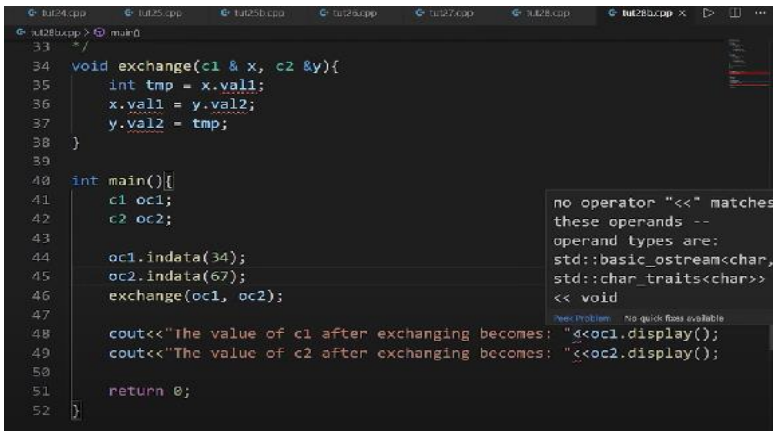


Also friendship in programming is not mutual and

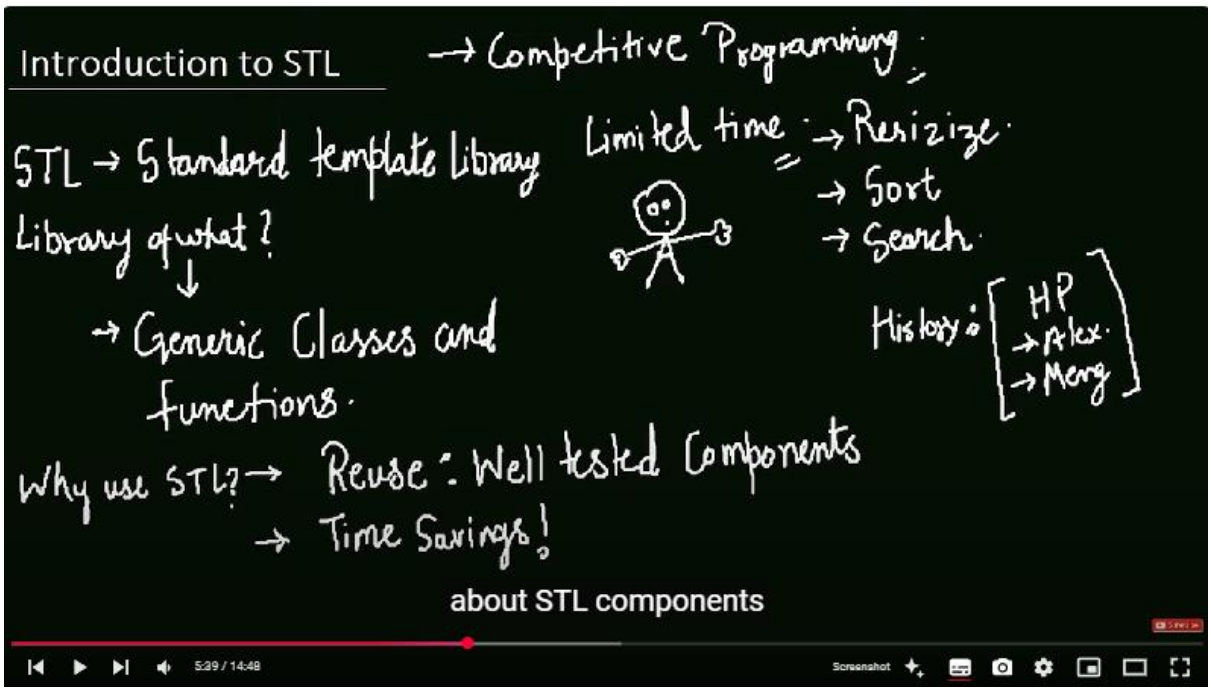
Friend is not bydefault inherited

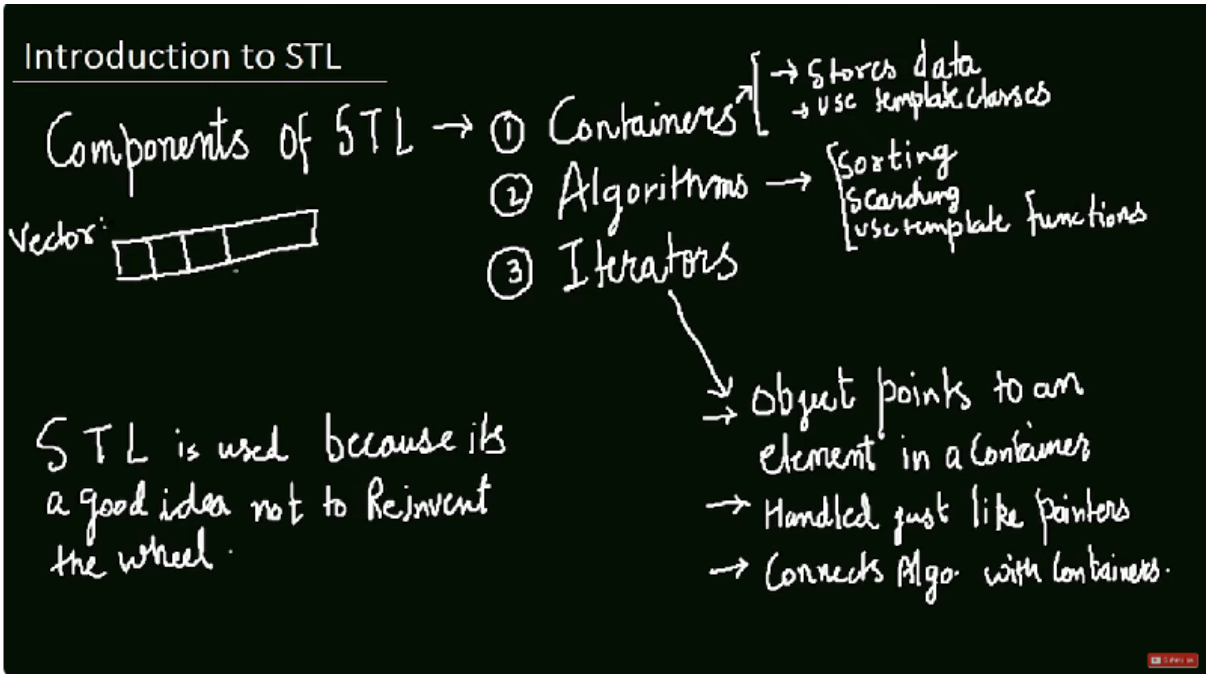
Used in operator overloading

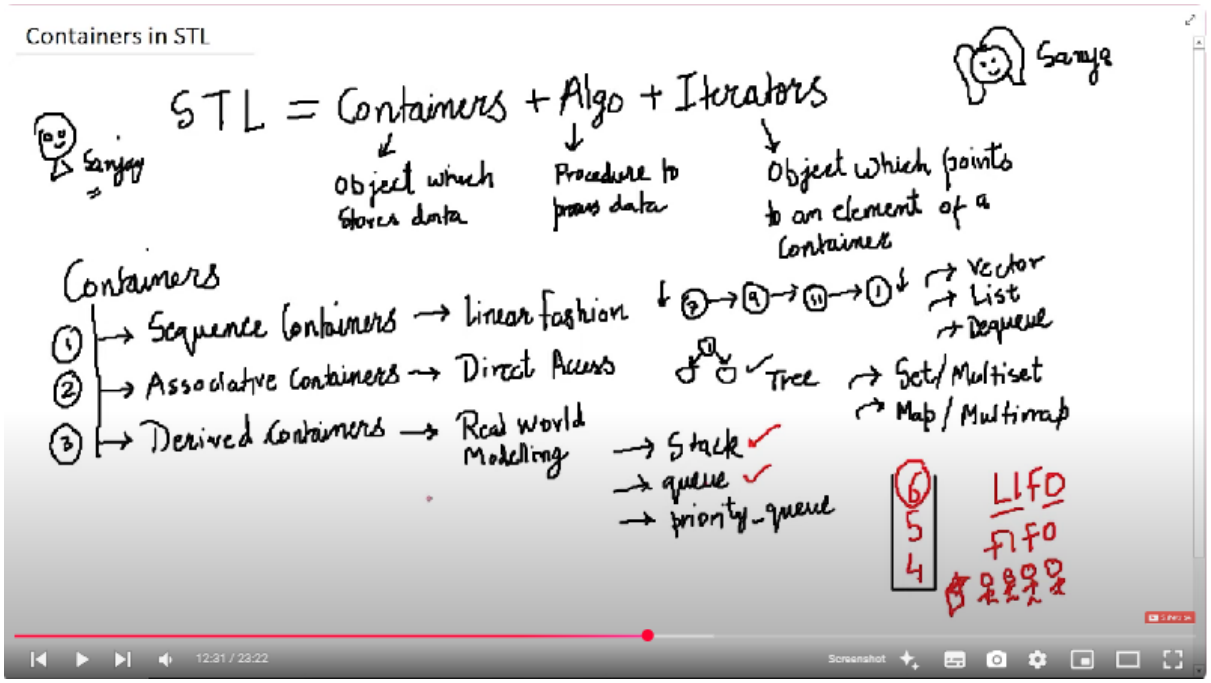
All members of struct are public by default and private in class

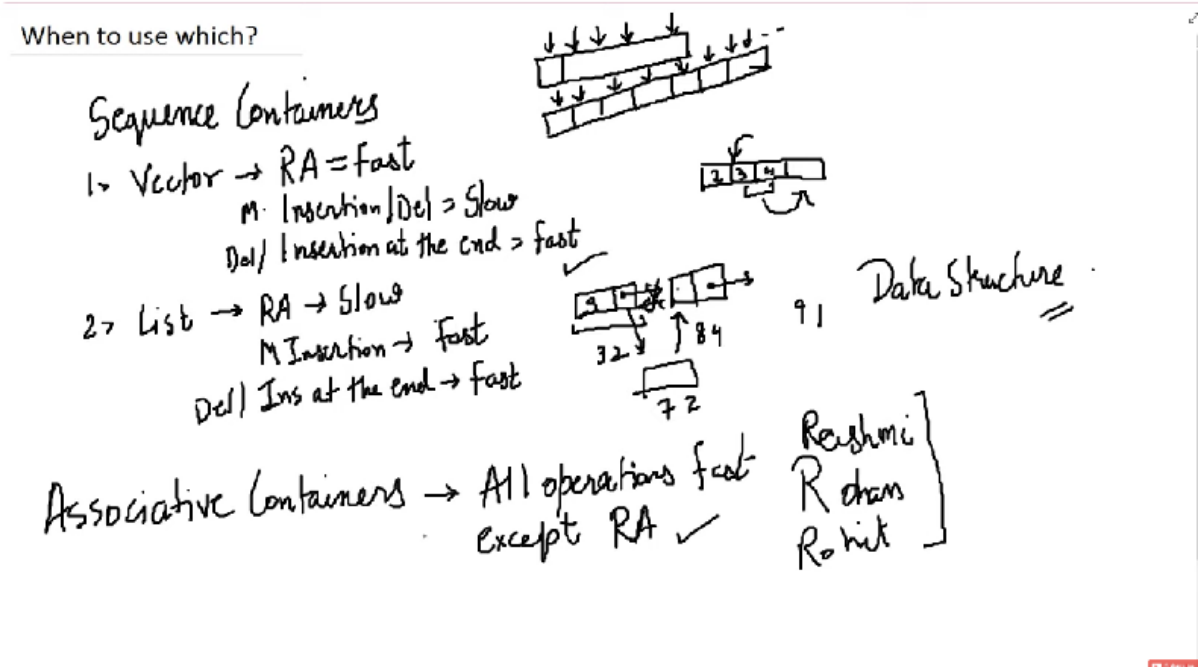


STL


Introduction to STL
→ Competitive Programming
STL → Standard template library
Library of what?
→ Generic Classes and functions.
Limited time → Resizing
→ Sort
→ Search
History: HP → Alex → Merg
Why use STL? → Reuse: Well tested Components
→ Time Savings!
about STL components
5:39 / 14:48
Screenshot

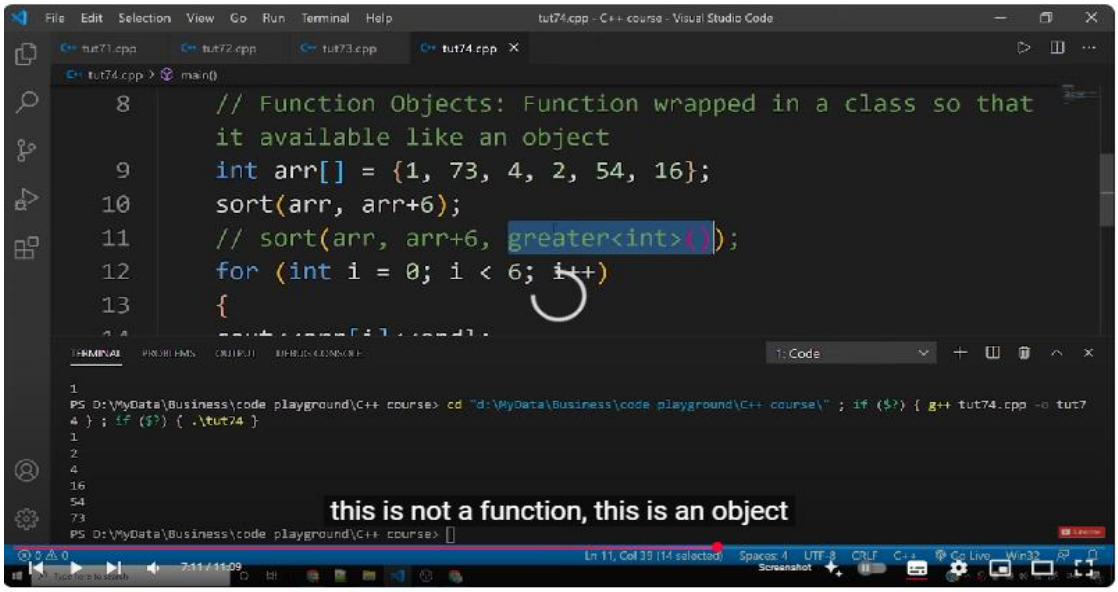

Introduction to STL
Components of STL → ① Containers → Stores data, use template classes
② Algorithms → Sorting, Searching, use template functions
③ Iterators
Vector:
→ object points to an element in a Container
→ Handled just like pointers
→ Connects Algo. with Containers.
STL is used because its a good idea not to Reinvent the wheel.





https://www.geeksforgeeks.org/cpp-stl-interview-questions/?ref=lbp

Functors



Function Objects (Functors) In C++ STL | C++ Tutorials for Beginners #74

A functor (or function object) is a C++ class that acts like a function. Functors are called using the same old function call syntax. To create a functor, we create a object that overloads the *operator()*.

The line,

```
MyFunctor(10);
```

Is same as

```
MyFunctor.operator()(10);
```

Let's dolve deeper and understand how this can actually be used in conjunction with STLs.

CPP

```cpp
// C++ program to demonstrate working of
// functors.
#include <bits/stdc++.h>
using namespace std;

// A functor
class increment
{
private:
    int num;
public:
    increment(int n) : num(n) { }

    // This operator overloading enables calling
    // operator function () on objects of increment
    int operator () (int arr_num) const {
        return num + arr_num;
    }
};

// Driver code
int main()
{
    int arr[] = {1, 2, 3, 4, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    int to_add = 5;

    transform(arr, arr+n, arr, increment(to_add));

    for (int i=0; i<n; i++)
        cout << arr[i] << " ";
}
```

Output:

```
6 7 8 9 10
```

Time Complexity: O(n)

Space Complexity: O(n) where n is the size of the array.

Smart pointer aur arrow function

Dangling pointer wagera

Virtual base

https://www.geeksforgeeks.org/cpp-pointers/

https://www.geeksforgeeks.org/cpp-interview-questions/?ref=lbp

https://www.geeksforgeeks.org/cpp-stl-interview-questions/?ref=lbp

https://www.geeksforgeeks.org/oops-interview-questions/?ref=lbp

https://www.geeksforgeeks.org/cpp-interview-questions/?ref=lbp

https://www.geeksforgeeks.org/smart-pointers-cpp/?ref=lbp