FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture18

Lecture 18: 'this' keyword in C++

Welcome to Lecture 18: Polymorphism. So, in the last lecture, if you recall, we saw this program to multiply two matrices using operator overloading, right. So, if you see, we have a private member data rows and columns, and then you have used a pointer to pointer, right. Then we have

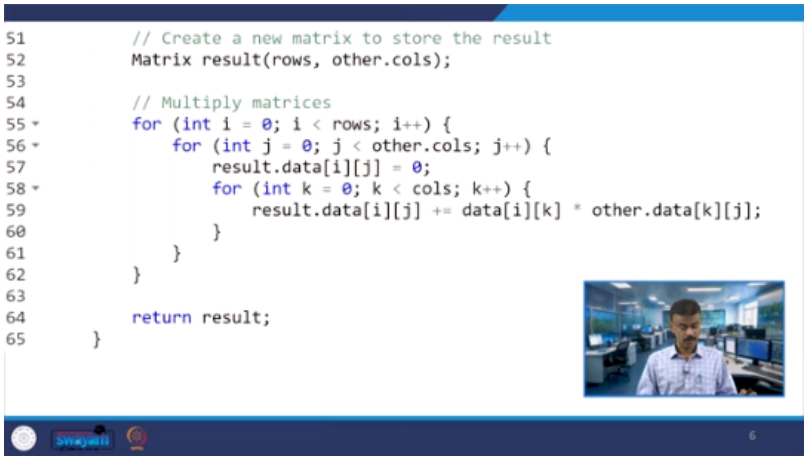
the constructor matrix(int R, int C), right. So, whatever we are passing, it will be assigned to rows and columns, right? And this is the dynamic memory allocation for a two-dimensional array, and then we have taken the user input, right? That is called the input matrix. So, for the rows and columns, what are the data that you are going to take from the user? And here, you have display matrix, all right. So, suppose we are getting, let us say, multiplication of two matrices or addition of two matrices, or even the matrix that you have taken as an input, how are you going to display that, all right.

So, that we have written from line number 33 to 42, and here you go, the operator overloading, right. So, it is like you have the name of the class matrix, and then it looks like a function, right? Operator star, and then you have inside, you have the reference for the matrix, right. Other is the reference for the matrix. It is like, if you recall, we wrote a program for complex addition or complex multiplication, or even both, right.

So, in the object we have taken, and then another one you are passing, right. The one object will invoke, and another one will be passed, right. So, that is what is happening in the main. In a similar way, here you have So, if you look at line number 45, columns not equal to other dot rows, all right.

So then these things will be done, right? If it is not right, suppose if you want to do the matrix multiplication, the column should be equal to the other matrix row, right. So, that you are checking the condition, and then here you go, the matrix

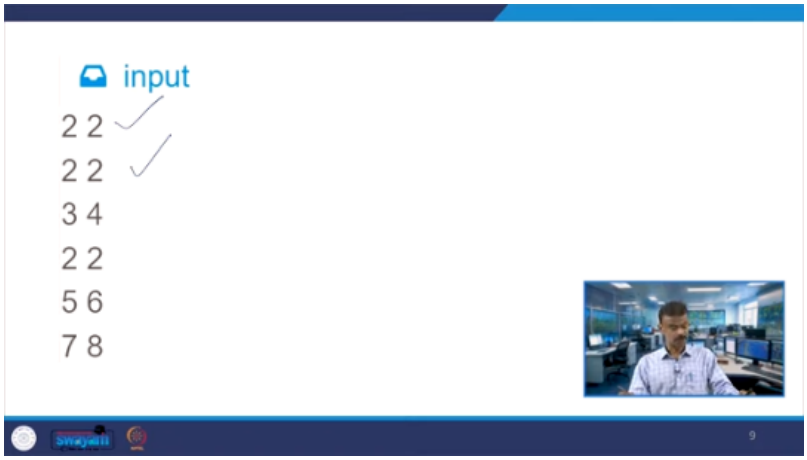
multiplication, all right. So, I runs from 0 to rows, and J runs from 0 to columns. And then initially you put data of ij is 0, and then for k is equal to 0, k less than columns, so we are multiplying the row and column component-wise, and then you are doing the addition, correct. So, when we take the input from the user, let us say we are taking 2 inputs from the user, 2 pairs of inputs from the user, that is R1 C1 and R2 C2, row 1 column 1, row 2 column 2, and then so here



you have matrix 1 object and matrix 2 object. So, then this matrix 1 object input matrix is calling. So, giving as an input. Matrix 2 is invoking input matrix. You are giving matrix 2 as an input.

And then, if you look at line number 95, So, here the star is acting as an operator overloading. So, usually, we use the star for multiplication. If there are two numbers, a star b will give you the multiplication of two numbers. So, here you are multiplying two matrices and putting in that result, okay.

The resultant one, the result is a matrix; you can see the class, right? So, let us see if you are giving 2 by 2, one 2 by 2 matrix as an input, another 2 by 2 as an input. So, the column and row, the column of the first matrix and the row of the second matrix, they are the same. So, these are the inputs.



This is one matrix. This is another matrix, right? When we multiply these two, right? At least, we can check what the first one we will get is. So, 3 into 5, we will get 15, right?

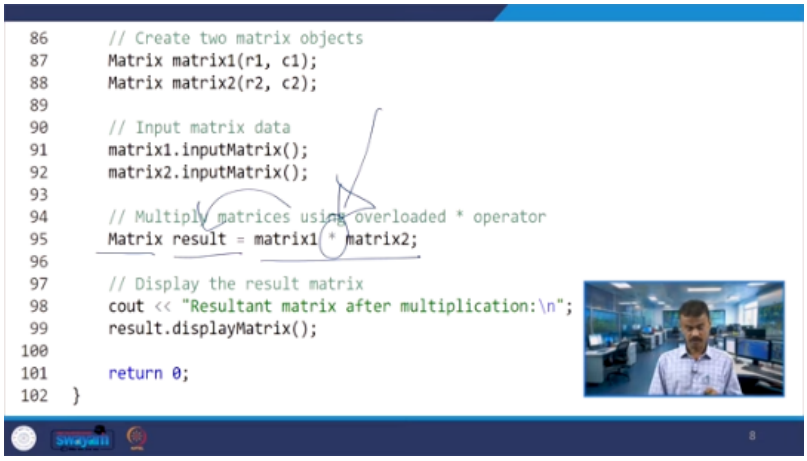
And 4 into 7 is 28, right? So the first element, you should get 43, right? A00, that result 00, you should get 43, right? So the rest you can check. So when you run the code, we will get the output.

So 15 plus 28, the first element was 43, right? So this is the resultant matrix after multiplication. So this works based on operator overloading. In fact, we had seen this program in the last class itself. I just, we have brushed it.

Right. So, matrix 1 star matrix 2 meaning is this multiplication; we have overloaded the star operator, right. Star operator we have overloaded, and we got the result, right. So, here usually star operators are for simple multiplication. So, here we have used this for matrix multiplication.

So, therefore, this is called operator overloading. We have seen function overloading, and this is an example of operator overloading. So now, we use this keyword, right? So this is a pointer. In C++, this plays a vital role, right? In fact, we can use it for multiple uses, right? So we can use this for multiple use cases, right? And we can call this a pointer. This is nothing but the 'this' keyword. When we use this, it is nothing but a pointer. So, this pointer is used as an implicit object parameter. So, when you are assigning, let us say, age, where you are passing age, some number, right? Salary, some number, or string, some string value, right. When you are doing this, right? When you are having this object's

member function, then when we are calling this, let us say, age, string, or year, right.



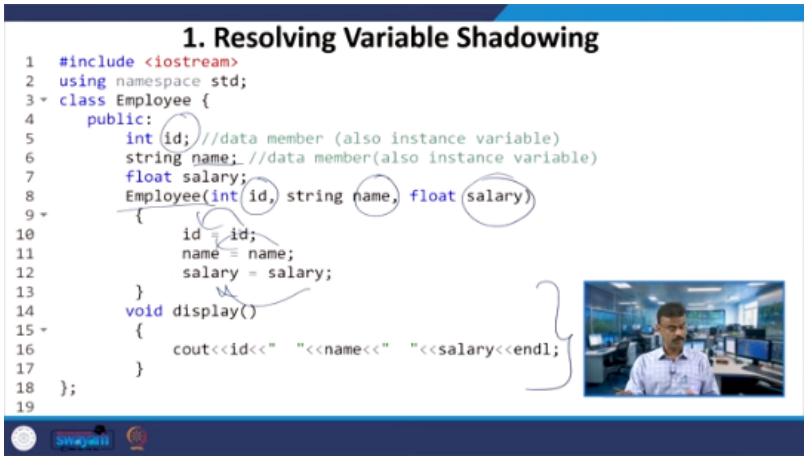
So, in that case, we use this pointer, right? Which is used as an implicit object parameter. When an object's member function is called, right? And it refers to the invoking object. So this may be, let us say, the instances or the member functions, right? So this dot, the member function.

Or we are going to see this arrow, the member function. So, those things are possible. So, when we consider three different cases. In fact, we have used this, and I told you that we would explain about this keyword slightly later. In this class, we are seeing about this keyword. So, now this keyword is useful for resolving variable shadowing.

So, what do you mean by variable shadowing? So, for example, consider this particular program, all right. So, consider this particular program. You have a class Employee, and then you have a data member called ID, and then you have a string name, right? Name, string name. So, then you have float salary. So, now suppose, right, you are passing, in fact, it is the parameterized constructor.

You have, let us say, ID, name, and salary. This is what I mentioned when we talked about the theory of what this is. So, here we assume that we are assigning ID to ID, name to name, and salary to salary. Right. And then you have one member function called display.

Right. So, now let us see what is happening. Right. Your class is over here. You have a constructor, and here you have the member function.



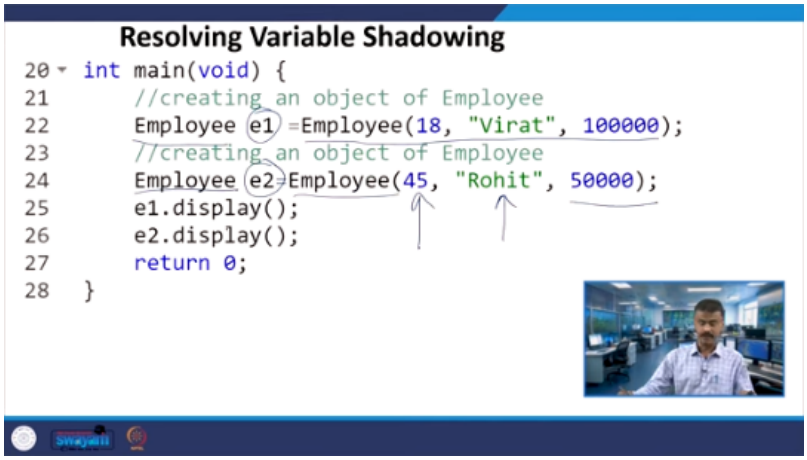
So, now you are creating, let us say, two objects. One object, E1, under employee, right? So, you are having the constructor; it is calling the constructor, which is the parameterized constructor, by passing the ID 18, name Virat, and 100,000 as the salary, all right? So, ID, name, and salary, right? So, 18, Virat, and 100,000 as the salary.

And the second object you have, E2, which is a Object whose class is employee, right? So, here you have the constructor, correct? So, 45 is the ID, Rohit is the name, and 50,000 is the salary, right? So, this you are passing.

So, that means in line number 22, you are passing 18, Virat, and 100,000, right? So, what we expect is ID will be assigned to ID, name will be assigned to name, and salary will be assigned to salary, right? So, we have passed 18, Virat, and the salary is 100,000. And in line number 24, you have passed 45, Rohit, and 50,000.

Right. So, 45 is nothing but the ID. Rohit is a name, and 50,000 is a salary. Right. So now, let us display.

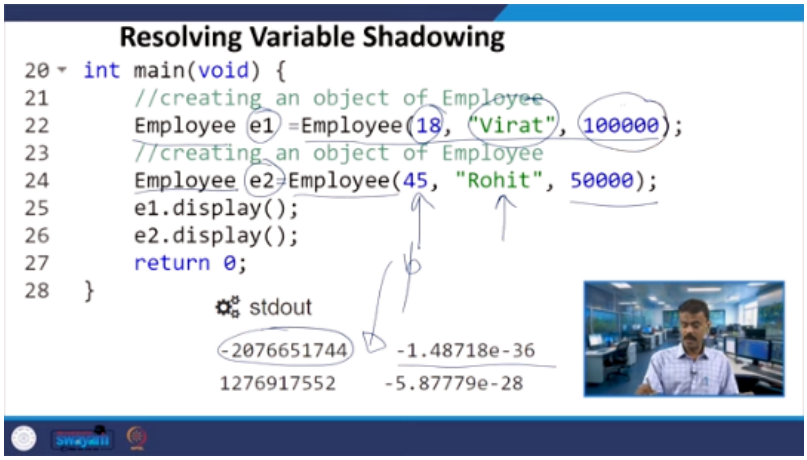
The E1 is invoking. Let us say, display. What is the output you will get? So, you expect 18, Virat, and 100,000. Right.



And on line number 26, E2 is invoking display. And you expect 45, Rohit, and 50,000. Right. So, this is the output we expect. But unfortunately, we get the output like this.

Right. It is showing some random number. Right. Garbage value, and the string is null, the blank space. And then the salary is this for E1.

Right. When E1 is invoking. But whereas we were expecting 18 Vroth and 100,000. And in the second case, also some random value, garbage value. The string is blank, and then here you have some random value.



How can you resolve this problem? So this is called resolving variable shadowing. So variables are becoming random. Right? So, there is a shadow, all right?

So, the ID you are assigning to the ID, you are expecting, all right? The ID will be printed, you are expecting the name will be printed, you are expecting the salary

will be printed, all right? But how to resolve this problem? So, that is why this pointer is playing a role. So, now with the help of this, all right?

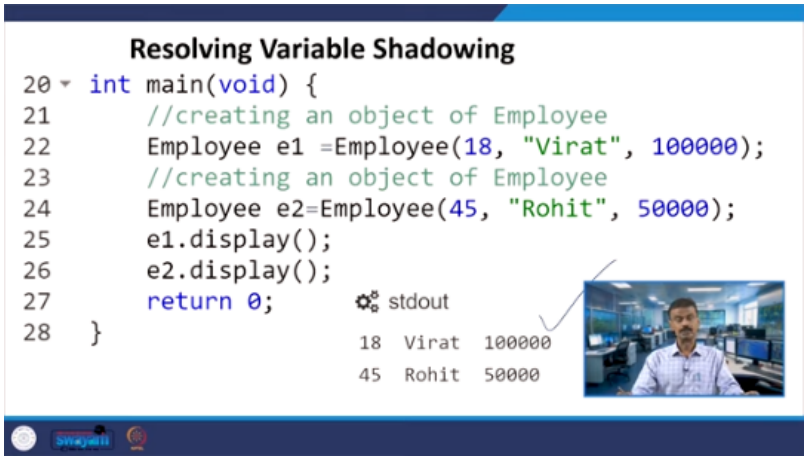
So, one more operator here you are seeing, the arrow operator, right? So, that Slightly later, I will come, I will talk about this arrow operator. And here you look, right, the same program, the ID is assigned to this arrow ID, right. So, because here we have the option for the pointer, the usual pointer concept in C++.

So, we are having this arrow operator, right. So, the name is assigned to this arrow name. And similarly, the salary is assigned to this arrow salary. Right? So, you are resolving the problem by writing this arrow.

Right? I hope it is clear for everyone. This arrow ID, this arrow name, and this arrow salary. Right? So, when you do this, now let your E1.

Right? The rest are the same. Whatever I explained, the program, all the lines, line number 1 to 28, are the same except for line numbers 10, 11, and 12. So, you are putting this arrow right.

So, what is happening when E1 is invoking display, E2 is invoking display. So, now you can see you are getting the output. So, 18 Vira and 100000 as output perfectly right, and 45 Rohit 50000 you are getting. So, this pointer variable, right, this is called a pointer, which is useful when you are assigning the variable, right, or the member data, or whatever the parameters that you are passing, right? So, it is useful when you are using this arrow, right? So, when you are using this and then you have an arrow, and then you are having ID equal to, right? Okay, so when I am running the code, I am getting the output like this, right? So, is it clear? Suppose if you do not put this arrow, you will get the garbage value. So, this concept is called variable shadowing. So, using this concept, I mean to say this



Arrow concept or this pointer concept, we can resolve this problem. Let us see how you are accessing the members using this, right. So again, one more case, right. So you have class test, right. So under this, you have one data member or member data test string whose data type is string, and here you have one member function, a non-static member function, or let us assume that.

Right now, it is a member function. In fact, we talked about static, right? You know static. So, non-static member function. And then here you have, you are passing the test string.

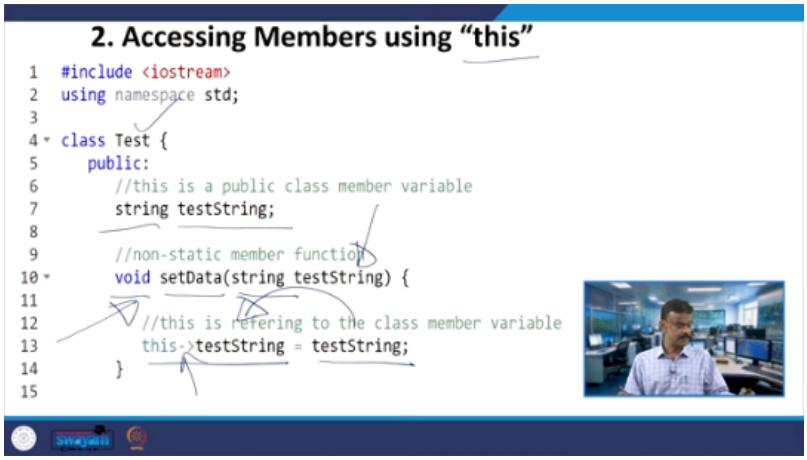
O is data type string, and this is a void function. So now, test string is assigned to this arrow test string. Like exactly what we have done in the previous program. So now you have get and print, right? Which is a member function.

Which is a void member function. Alright. And then you have string str. Correct. So its data type is string.

Now again, line number 18. First time we are seeing it. This arrow sets data of str. Right. So set data is the member function.

Right. So set data is the member function. So here you go. This is a set data member function. So under set data, we are again using this arrow.

Alright. So now you can see what is happening. So this arrow sets data of str, this you are passing, right? This you are passing, and then we will see how this is working, right? By taking the because this is nothing but the member function. So here what you are doing, you have one cout statement: the string is this arrow test string. So three places we are using this arrow, right?



So now When your class is over up to here, now go to the main program. Main program, you have the test object, correct? Whose class is Test, capital T, right? Test is of small t is an object, and capital T is a class. So now this test object, right? This test object which is invoking get and print, right? So when it is invoking get and print, right? When it is invoking get and print. So here you are passing this. This is a test for member accession. All right.

This is a test for member accession. This is a test for member accession. And that is a string that I am passing. Right. So when you are passing this string, that is used in line number 18.

So set data str. So when you use simply, it will be null. So here this arrow set data of str. So here you go. Set data of test string.

All right. So test string is nothing but now this one. This is a test for member accession. Correct. So you are this arrow test string is nothing but now this is a test for member accession.

Correct. So now you are in line number 19. So line number 19, the string is, right? What is the string? This is a test for member accession.

That means this arrow string is nothing but this, right? So this is what we are assigning over here, correct? This is what we are assigning over here, whatever the string is. So this is the string I am passing, correct? And then this will be printed because of this, right?

So when I run the code, I will get the output. The string is, right? So this is the string that is printing, right? With a colon, and then this is a test for member accession that will be printed. Right.

So this is how this keyword works. So the case of member data. Right. So this is a case of member data. Correct.

So it is infant member data. Right. Or the parameter that you are passing. And one is for how you are calling the function. SetData is a member function, in fact, right?

In the previous program, we had seen how we can call whatever the parameter value that you are passing, right? How to assign? How this arrow is useful, right? So, to print or store those values that you are passing, right? So, in a similar way, we can call the member function also.

SetData is a member function, and this arrow is So when you have set data like this, you will get the output like this: the string is 'This is a test for member accession,' right. So, like this, we can use this, so that we have seen two different cases, right? How this can be used for the parameters or the variables and how this can be accessing the member function as well. So, the next one we are going to see is the pointer, right.

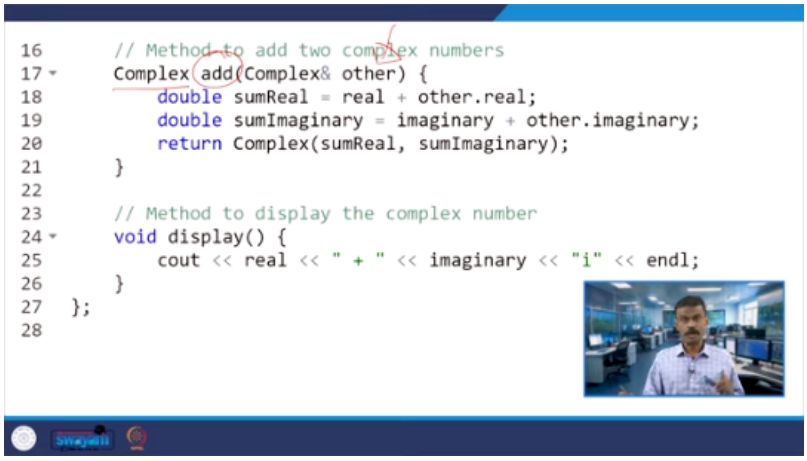
So here, when you look at the class Complex, we have two private member data: real and imaginary, right. So then here you go, the constructor Complex, right? So where you have R and I, we are going to pass. Right? And then that will be assigned to real and imaginary. Right?

So that is the constructor. And then here you have one member function called add, whose class is Complex. Right? So you are passing one complex number. Correct?

So you are passing one complex number. Right? So here you have real plus other dot real. So you are adding. So, in the main program, what happens?

So, one object will invoke this add function and another complex number you are passing, all right. So, whichever is invoking those real and imaginary is this, and since you are passing another complex number, the other complex number's real and imaginary is this, all right. So, then it is returning the complex number with a

real and imaginary, and here you have the display member function. So, that will print it is of the form a plus ib, right.



So, which will be printed as 2 plus 3i, 4 plus pi i, something like that, right? So, that will be displayed over here. So now you have user real and user imaginary double value. I mean, these two are variable names, right?

So, you are taking the input from the user and you are creating one object. User complex is an object, right? So, you are creating one complex number. So, you have taken the input from the user. For example, I am taking, let us say, 2, 3, right?

If at all, I take. Alright, user real is 2, user imaginary is 3. Alright, so here you have the complex, which is the class, and user complex is the object. Alright, another one is a constructor complex. So, the constructor complex has 3.5 real, 2.0 imaginary.

Alright, and complex is the class. Alright, that means the constructor complex. So, which is another object, another complex number. Alright, So, one you are taking the user, right, one input you are taking from the user, another you are passing, right, using the constructor.

So, when I am passing this, let us say 3.5 and 2.0. So, in fact, here also whatever the input that I am taking from the user, I am passing. So, that means this constructor will be called or will be assigned to real and I will be assigned to imaginary, right. Similarly, here it is happening 3.5, 2.0. The real and respective imaginary will be assigned for the complex number, this particular complex number, right, that means this object constructor complex.

So now you have the result complex number or the result object whose class is complex, and user complex, right? This user complex here, you go, this is a user complex. This is invoking the add function by passing constructor complex. So assume that this is 2, 3, I, Z. Assume that. So that means 2 plus 3, I, we are passing, right? In fact, this user complex contains, right? And this is invoking.

And the constructor complex, we are passing. 3.5 plus 2.0, I, we are passing. So it is equivalent to that, right? For the understanding. So that means user complex is invoking the add function.

And then you are passing this 3.5 plus 2.0, I. All right. In the constructor complex. This is what exactly happening. All right.

So now see our user's complex number. So user complex number dot display. All right. So display what is happening. The first number whatever you are taking from the user.

So that will be printed. Similarly, constructor complex will invoke display. All right. So then whatever you have passed, 3.5 plus 2.0 will be printed. All right.

So see out result dot display. So this, in fact, we had seen, right? So when you have 5 and 2 from the user input, the first complex number will be 5 plus 2i, right? That is nothing but the user complex number. And the second one is a constructor, 3.5 plus 2.5.

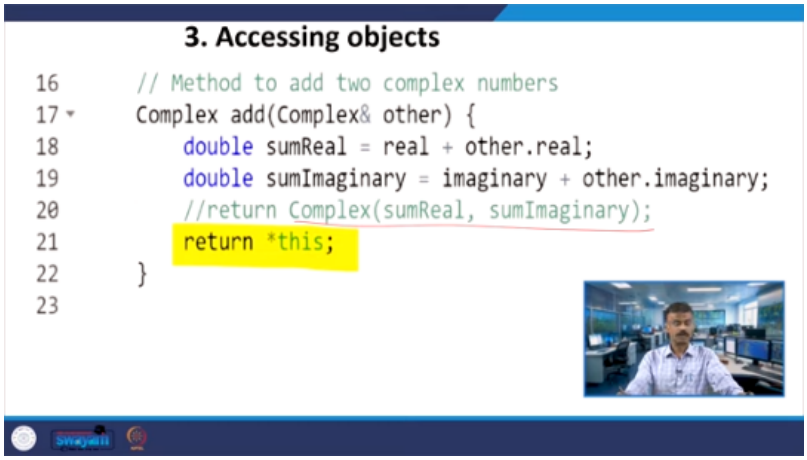
The addition of these two is 8.5 plus 4i. So now the question is, where exactly are we using this? Right. So here you go. So when you want to access the object.

So as I said, one of the objects is invoking the function where the other object is being passed. Right. So, suppose. So here, in this case. If I want to display.

Right. The resultant of the complex number. We simply put return complex. Some real. Some imaginary.

Right. So, we are returning this. Or. You can have some complex. Result.

Right. Store it in the result. So, that will also be displayed. Suppose I want to display other.real and other.imaginary. What can I do?



Simply return other. Right. Now, the question is, suppose I want to display this. That's right. This real and this imaginary.

Right. Suppose I want to display this real and this imaginary. What can I do? So, I can use this pointer like this: return *this. So, when I use this return *this, this is the pointer.

In fact, we are using the dereference operator. So, addition is not correct here. So, here is how I am going to display the object that is invoking the function, where you are passing certain values through the object. All right. So now, the object that is invoking.

So, I want to display right inside the member function. So, for this, return *this is useful. So, that means I can access the object with the help of this. Right. I can access the object with the help of *this.

So now you see 5, 8 I am giving as an input. Right. 5, 8 I am giving as an input. And you can see that the first complex number is 5 plus 8i. And the constructor complex number is 3.5 plus 2i.

In fact, the addition of these two, we have done that. So, our case is this is not the sum of complex numbers. You forget about this. So, in fact, this should be the first complex number. I just put that return star this.

I have not disturbed the program. So, from the addition point of view, this is not right, but from the concept point of view, return star this is how I can access this object. So, I can access this object with the help of this, right. So, when I put this,

you can see that the object that is invoking, right. So, here you go, the main program, right, user complex that is invoking at line number 44.

User complex that is invoking the function add. So when it is invoking the function add, and then you are passing this. Now my question is, how to display this user complex inside that member function. So for accessing this object, I need star this pointer. Return star this.

So when I do this, 5 comma 8 I am passing. In fact, that particular object is invoking the add function, and then you are passing this constructor complex number, right? So you are getting, right? You are getting by putting return star this, right? By putting return star this, so that will be printing the first complex.

So constructor complex number, if I ask you to print, I mean which is nothing, right? So, what will you do? You simply put return other. So, when I put return other, we know that I will get 3.5 plus 2i. But how do you access that particular object which is invoking the function, right?

So, for this, you use return star this, and return star this will return the first complex number, right. So, this is how the this keyword works, right. So, the this keyword You can assign whatever parameters you are passing, all right. In fact, in the case of a constructor, when we saw the first example, right, that Virat and Rohit example, so how you are assigning the data or the variable, all right.

Even in some of the programs we previously saw, how you can assign with the help of this. In fact, we had seen this dot. If you recall, we had seen this dot. In the second program, We accessed the member data, right, accessed the member function, right, accessed the member function set data, this arrow set data, if you recall, right, and then whatever the string that you are passing, the set data with the help of again the this arrow operator, we would be able to handle, and here in this case, the addition of two complex numbers, so we are passing one complex number with the help of a constructor.

Another one is invoking the member function, right. So, one object is invoking the member function at, and you are passing this constructor complex number. So, in that case, right, I want to print the object that is invoking the member function. So, for accessing the objects, we have used return star this, right. So, in today's class, we have seen operator overloading.

And this keyword. Right. So this keyword. So three different cases. The member data.

Or simply data. Member function. And then how you can access the objects. Right. So with this, I am concluding.

And in the next class, we will talk about method overloading. Thank you.