FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

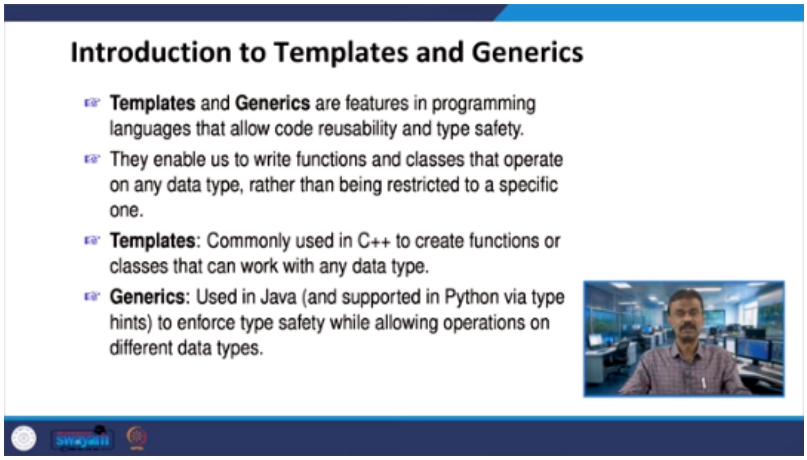
Lecture36

Lecture 36: Introduction to Templates and Generics

Thank you. Welcome to lecture 36, the new chapter called Templates and Generics. So, in this chapter, we are going to see what you mean by templates and generics. So, these are all features in programming languages such as C++ and Java, allowing code reusability and type safety. So, code reusability in the sense, for example, I want to write a program for the maximum of two numbers, all right.

So, you may have the maximum of two integers, or the maximum of two real numbers, right? Or the maximum of two characters also, right? So, in that case, suppose till now if I ask this program to you, you will write separate functions, right? So now the question is, can I write—because the logic is the same, right? The maximum of two numbers or two characters, correct? So in that case, can I use only one function, right? Or one class so that I can use it for finding out the maximum of two integers, the maximum of two doubles, or the maximum of two floats, or the maximum of two characters, and the maximum of two strings, right? So that is nothing but the code reusability and the type safety in the sense. Okay, suppose you are using, let us say, the maximum of two integers, right? Or the maximum of two doubles, right? So in that case, which data type we are going to use will be taken care of, and accordingly, the program will work, right? That is called type safety. So when I am talking about templates or generics, right? So, in the case of C++ or even in Java, we use them for functions and classes, right.

So, we can use a template class and template functions. So, these can operate on any data type, right? So, rather than being restricted to a specific one. So, commonly used concepts, right, in C++. So, we have templates, all right.



So, the templates are used to create functions or classes. So, which can work in Any data type. Right. Which can work with any data type.

And generics. Right. So this is being used in Java. Right. So this is being used in Java.

And this is supported in Python via type ints. Anyway we are going to see that. So, this is enforcing type safety. So, we already saw what is meant by type safety. So, the generics.

So, that will enforce type safety. So, that is allowing operations on different data types. So, let us see certain terminologies. So, when I define templates. So, first initially we go through what do you mean by templates.

So, template is nothing but a feature in C++. So, it is available in Java as well. So, this is a feature. So, you can write the functions and classes to operate with the generic types. So, when I am talking about the templates all right.

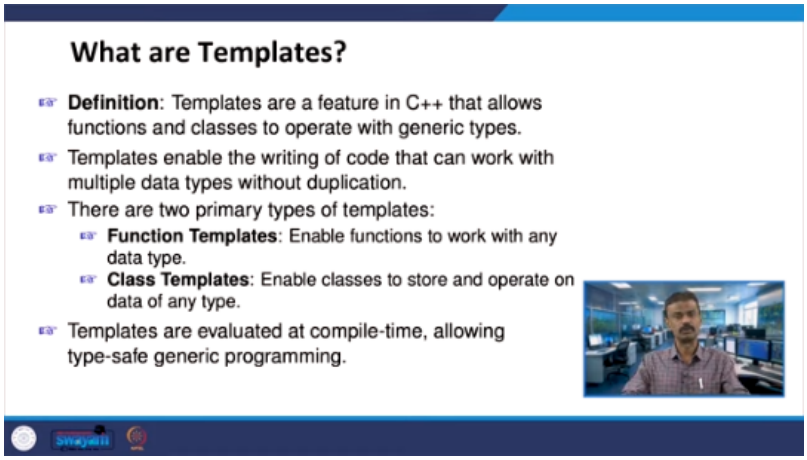
So, when when when you are writing the code. So, this will enable to write the code which can work with any data types or multiple data types without any duplications. As I already said, so we can have two primary types of templates. One is the function template, another one is a class template. So, the function templates which enables functions to work with any data type.

That is what I gave an example, maximum of two numbers, right. So, you may take a data type which may be integer, right, or which may be double, right, which may be float, right. So, that means you can work with any data type.

Similarly, class template. So the class template also, right, so this enable classes to store and operate on data of any type.

So here also you can have, we are going to see with the help of template keyword, I can use it for integer, I can use it for double, I can use it for character, etc. So when you use template, right, whether it is a function or class, so this can be evaluated at the compile time. that during compile time it will be evaluated and it is allowing type safe generic programming right. So, type safe in the sense when suppose you are using for let us say finding out the maximum of two integers. or maximum of two doubles right.

So, it will be taken care accordingly right. So, we are using we are going to use only one function with the template and then. So, any data type that you are using. So, which will be the type saves generic programming. So, what are all the advantages?

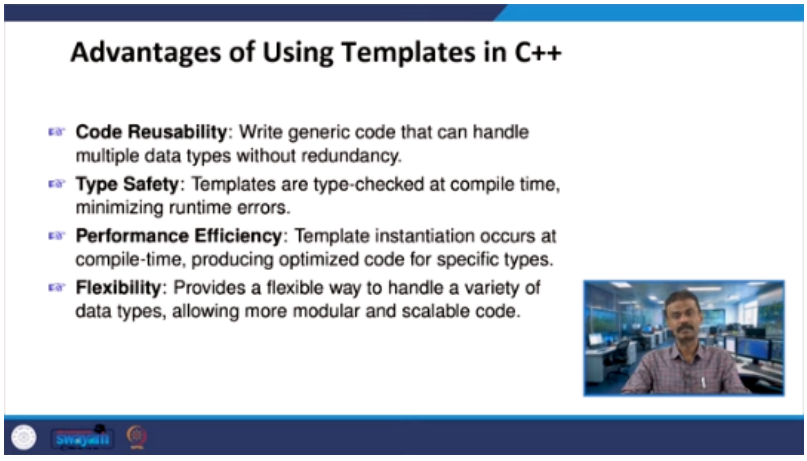


The code reusability right. So, this keyword we have already seen the code reusability. So, that is what I said maximum of two numbers I do not want to write it separately. for integers for doubles or float or characters all right. So, I can I can I am I can use only one template function and I can call whenever I want for different data types all right and then there would not be any redundancy type safety all right.

So, type safety so, since we have already seen the templates are type checked at compile time. right if suppose you are using let us say integer right maximum of two integers maximum of two doubles right so which will be taken care and since this is being type checked during compile time so this will minimize the

error during the runtime the third point is performance efficiency right obviously i mean suppose you are writing only one function I mean it is a efficient right and when you are doing the instantiation right the template instantiation also I said this is occurring during compile time. So, it will produce the optimized code for specific data types. So, like I mean I am taking the same example like the maximum of two integers maximum of two doubles right.

So, I am going to write only one function. So, when I instantiate this, when this is occurring during compile time, right, when I instantiate this, this is occurring during compile time and I will have a optimized code, right. For example, that four line code, maximum of two numbers, I can use it for integer, double or float, etc., right. And the last point is the flexibility, all right. The meaning of flexibility, I mean, you can handle any data types, right.



For example, the same case, If you are writing maximum of two numbers, I can handle integer or double, right, or character, etc., right. So, the flexible way to handle the variety of data types. And the code, when I am considering the code, so it is a modular and scalable. So, now we will see the syntax of template functions in C++.

So, this allows a single definition. So, when I am going to write a function, it will have a single definition. And I can use it for variety of data types. All right. So, for example, it goes like this template type name T. In some book you can find class T. Right.

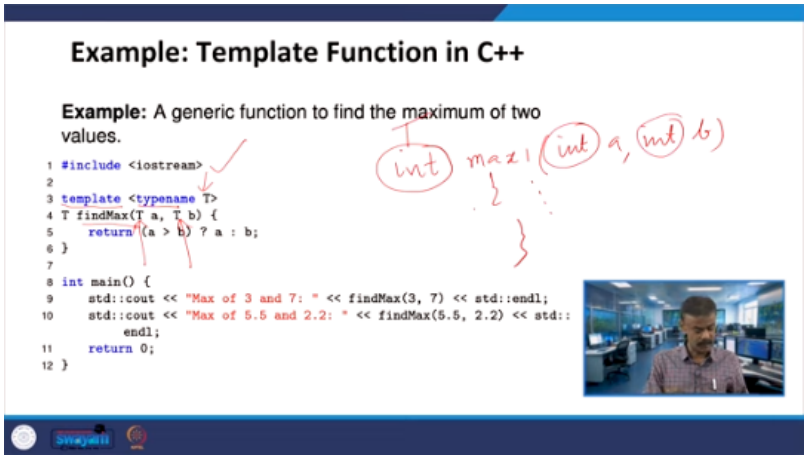
So, type name. And here you can see name of the function is function name. Right. And then you have two parameters. Two parameters also you put T. All right.

And the return type T. Right. For example, I want to consider an integer. Right. I want to consider an integer. So, in that case, when I am calling a function named T, it will be an integer. Or suppose I am going to take a double, right? T will be a double or a character.

So, this flexibility we have. So, in fact, we will see an example. This is what I said: maximum of two numbers. This is what I keep on telling—this example. So, let us find the maximum of two values.

One is the maximum of two integers. Another one is the maximum of two doubles or floating-point numbers. So, in this case, when I am writing the function, right, the syntax is—yeah, this you know: #include . Then you can see the syntax: template, use a keyword template, then you have typename, name of, typename is T. You can give any other name—U also you can give, T1 you can give. It is like an identifier, all right. And find max, you are putting T, a variable a, T, variable b. Suppose I am—I mean, I am asking you to write a function to find the maximum of two integers.

How did we write? So, you put the return type integer maximum, let us say max1(int a, int b). This is what we wrote so far, and then you write the logic, right? So, now if you look, I replace all int by t. This is what the syntax: t max1. So, here we are having findMax, and here you put t t. So, now once you put this t, that means the type name. You can use the same function for double.



You can use the same function for float, correct? Even a mixture also, we are going to see. Some examples we are going to see, the mixture, all right? So, the logic you know, right? The logic you know, whichever is the maximum, that will be maximum, right?

Return A greater than B, right? If it is true, it will return A. If it is false, it will return B, all right? So, this we know. So now you go over here. I am going to find out the maximum of 3 and 7.

So, what do you do? You call the function. Find max. Template function. You are calling the template function.

Find max of 3, 7. Right. So, you are passing 3 and 7. We know 3 and 7 are integers. Now, the type name will automatically change to integer.

This is the advantage. This is what I kept telling. Right. So, it is handling. And then safety also.

So because I am using integer. Yes. It will be converting everything to integer. So, that means this t, this is also integer, this is also integer, this is also integer. So, all will be integer.

I hope you understood. Now, that is all the simple function you know, right. Once the t is becoming integer, maximum of 2, the logic is if a greater than b, a is the maximum, else b is the maximum. That is what we have written in line number 5, right. So, that output you will get it, endel.

Next again, I am calling for 5.5 and 2.2. So, when I have 5.5 and 2.2 you know that both are double. So, when both are double 5.5 is double it is calling the function find max you are passing 5.5 and 2.2 right. So, now the t will be automatically double right.

So, it will be automatically double. So, when T is double, this will also be a return type double. A is double, B is also double, all are double. Correct. So, then the logic is the same: 5.5 and 2.2, which is maximum. So, the first output I will get is 7, the second output I will get is 5.5. Maybe we will run the code. So, we will go to the C++ console, find max CPP. Yes, the same code that I explained, in fact the same, yes.

So, you can see that the maximum of 3 and 7 is 7, the maximum of 5.5 and 5.2 is 5.5. Okay. So, we got this. Now you may ask the question. If you look at line number 11.

So, find max of 5, 5.2. Right. So, there is a chance of throwing an error. In some compilers, what it does. The 5 will be considered as 5.0.

Right. So these kinds of test cases. You yourself can check. Right. Or let us consider that.

We will remove the common statement. And then we will run the code. Now we will see what it is doing. So you can see that there is an error. The reason is there is a mismatch.

One is integer, another one is double. So the template class cannot take this. The template type name T cannot take this. So when you are using line number 9, the data type is safe. When you use line number 10, there is no problem with the data type.

That is what we talked about the safety. But whereas here, the compiler is getting confused. Type name. What is the type name? Right.

Because one you are passing integer, another one you are passing double. Right. You can check with one character and one double. Right. So you should get an error.

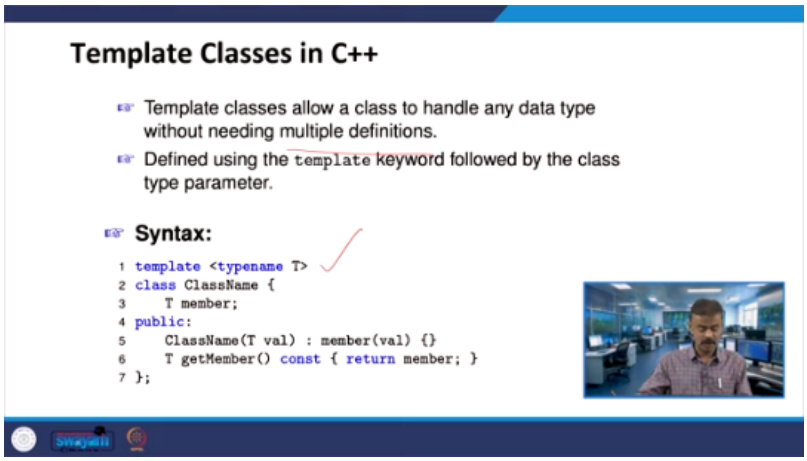
So this will throw an error. Otherwise, we had seen suppose I commented it because we know that it will give an error. So, we have run the code and we got

the output. I hope you are slowly understanding what do you mean by template function right. So, this is what we have done.

So, in the both the cases So, first case it is taking considering t as a integer and second case it is considering t as double. So, next concept is template classes in C++. So, we had seen template function and this is a syntax for template classes, right. So, when I am talking about template classes, it is allowing a class to handle any data type, right.

So, similarly we put the t, right. So, t it may be a integer, it may be a double, etc., So we no need to have the multiple definitions. For example, I have a template type name T, this example. So we use the same keyword template followed by the class type parameter.

So assume that I have template, let us say type name T. And then here you have class name of the class is class name. And inside I use a member data t. It can be integer. It can be a double. But once you are using let us say double throughout it should be double.



Otherwise, like a function, we got the error right. So, you will get an error. So, assume that T starts with an integer. So here, line number 3, T should be an integer. And here, you have one constructor, right, which is a class name constructor.

So here, it should be an integer if it is an integer. If T is an integer, right, and also you have a member function, get member, right, which is returning a member, and the return type should also be an integer, all right. If you are considering the whole T as an integer, or whole T you want to consider as a double, right, or

whole T is one of the data types. So, you have to be a little careful, okay. So, for example, in line number 5, if you try to use a val, right, the value which is a double,

then it will throw an error. Assume that you are passing something, or when you are creating an object, right, you are passing with the double, then it will throw an error, whereas the member T you are considering, right, the type name you are considering, is an integer. So, when I am considering it as one data type, integer, throughout it should be an integer, or if I am considering T as a double, throughout it should be a double. So, this you keep in mind, and based on that, we will see this particular program. So, this is the case of the template class, right, a generic class to store and retrieve a value of any type, right. So, let us consider template type name T, class storage, right, you have a

one member data value whose data type is t, t is type name and here I have a constructor storage. So, when I am creating an object you have to pass the value should be assume that if it is a integer throughout the t should be integer right. So, whatever you are passing will be copied into value that is a meaning right. So, this is the parametrized constructor and you have one member function get value whose return type is t and it is returning value okay so this is class storage so now in the main program so assume that i have an object in storage right so when you are writing the class name you can see first time we are seeing

Line number 12, storage. We know it is a class name. And then you are writing a type name. So, what is the type name? Integer.

Correct. And you are passing 100. You should pass the integer. So, suppose here you are passing, let us say, 100.23. Right?

You will get an error. Right? Because t is an integer. Here you are specifying it is an integer. This is a text.

So far, we have not seen this. Right? Because template... You are writing type name t. So, the type name you have to mention when you are instantiating an object. So, here we have instantiated an object in storage and you are passing the value 100.

So, when you are passing 100, it is nothing but val is equal to 100 and all t will become integer, right? So, value equal to val, right? So, that is what you are

doing in the constructor. So, now value is becoming 100 and similarly you have another storage, right? The class storage whose data type is double, whose type name is double. So in that case, all t will be replaced by double and then you are having one object called double storage by passing 99.99. So we know that 99.99 is a double, correct? So now this value for this object, right? Value equal to val. Val is you are passing 99.99.

So, your value will become 99.99 for the case of object double storage. So, now see out integer storage that in storage object this particular object will invoke get value. So, when it is invoking get value, line number 8 you have get value, it is returning value, right. So, in the case of first object, you are passing 100. So, here you have to get 100, correct.

So, return value which is 100. And in the case of a double storage, double storage is invoking get value, right. We have passed 99.99. So, you should get 99.99, right. So, these are the output you will get.

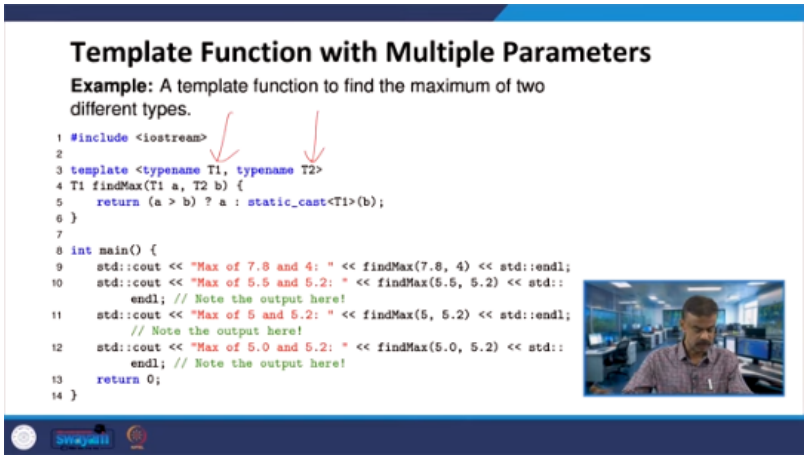
So, when we run the code you can see that we will go to C++ the same example we took. So, when we compile and run here you can see integer storage you are getting the output 100 and double storage you are getting the output 99.99. right so this is what I hope you passed just press the key and we are going yes you have passed 100 and 99.99 so you are getting the output. So now you have another example for class so you have understood one we have taken for function and this one we are taking for the class i hope you understood all right so when i put the type name t the type name t can be used for any data type all right so this is a anyway i have explained this so you have storage store is name of the class storage which is a template class and it is storing a value of any type right t so here in this example we use integer and double and when the class is instantiated

Let us say with specific types integer or double. So, this will be happening at runtime. So, now let us have the function with multiple parameters all right. Let us have a function template function with multiple parameters. So, type name t 1 and another one type name t 2 correct.

So, if you use this we have got a error if you remember maximum of one integer one double right. So, now how to address those case? So what you can do, you

can use whatever type names you want. So here we have used type name t1 and type name t2 in line number 3. And then we are going to run the same code.

Let us say find max, right? Find max of t1 a and t2 b. So one type is t1, another type is t2. So that I can handle, let us say float and integer or double an integer, right? So a greater than b. So it will return a. Or it is typecasting to T1.



Right. Typecasting to T1. So either you can do it here or before A you do this. Typecast. Right.

Static cast. Let us say T2. Right. So you can do this. So now you see you are calling FindMax.

7.8 comma 4. So one is double. Another one is integer. Right. So here in this case what will happen.

So since this one is 7.8 and another one is 4. And we know that 7.8 is greater than 4. So even without typecasting, this code should work because you have specified the type names T1 and T2. One is double, another one is integer, right? Even assuming that it is getting typecasted, fine.

It will become 4.0, all right? So the maximum of 7.8 and 4.0. So this will print 7.8. And the second case, find the maximum of 5.5 and 5.2. Both are double, right?

So here also, typecasting is not required. Both are double, correct? So even if you do that, double will remain double. So, 5.5 and 5.2, we know that 5.5 is bigger, right? Maximum.

And the next one is interesting, right? So, you have find maximum of one is integer, another one is double, right? So, when you type cast this static underscore cast to t1, there will be loss of data, right? Precision will be lost. So, find max of 5 comma 5 and then you may get 5.

We will check that right you will get 5 and the last one is 5.0 and 5.2. So, this is similar to the one we had seen in line number 10 both are double. So, 5.2 is bigger. So, these are the output we will get. So, we will run the code and check right.

So, here you go. So, we took 7.84. 5.5, 5.2, 5, 5.2, 5.0 and 5.2. So, let us run the code. Yes.

All right. So, the first one, yes, whatever we have expected is 7.8. The second one, both are double 5.5. And in the third case, t1 is integer, t2 is double. And when you are typecasting double to integer, it is saying the maximum of 5 and 5.

So, therefore, you are getting 5. So, you have to be a little careful. So, when you are doing or when you are using more than one type name, all right, and your logic. So, what you can do in this case is do it the other way around, right? So, when you go to the program.

So, you can see that return a greater than b, right? So, here you do the type cast static underscore cast t2. of A and here it is B, right? This problem will be solved. So, whichever is a higher type, right? So, you have to be very careful because you are converting into a lower type.

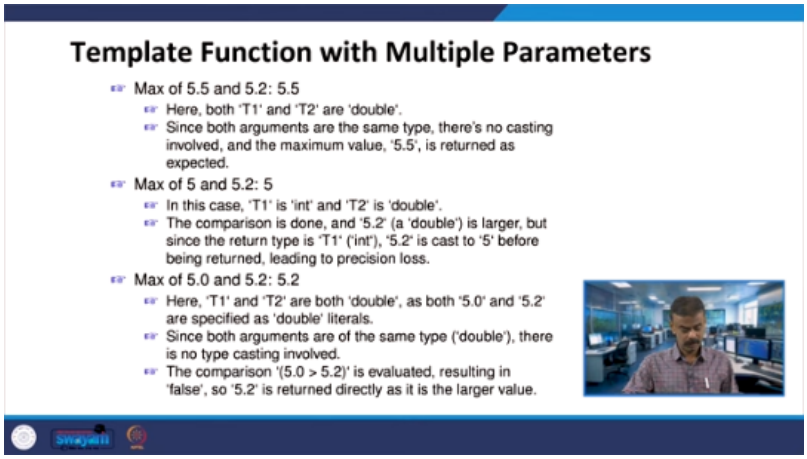
So, you may get the output like this. So, here in this case the maximum of 5 and 5.2 it is saying 5 which is not correct right. So, this is due to 5.2 is being converted to 5. So, then you are getting the error value right. So, you are getting 5 which is not correct I mean to say.

And line number 12, 5.0 and 5.2, you are getting 5.2. I hope it is clear. Now, even when I use different type names, the program is working fine. So, one is T1, another one is T2. So, I can use it for program is working fine, right.

We are not getting the right output. when you are giving 5 and 5.2 all right. So, one has to be very careful how to handle other three test cases trivial. So, we are getting the output right. So, this is what we expected the output and we also got it.

So, this is how it is working right. So, one case we are taking 7.8 and 4 one is double and another one is integer. So, even here in this case that is what even without right doing the typecasting. So, we know that 7.8 is greater than 4 and then we are getting the output 7.8 and similarly 5.5 and 5.2 both are double we are getting 5.5 and in the case of maximum of 5 and 5.2 there is a precision loss all right. So, 5.2 is converted into 5 and then maximum of 5 and 5 is 5 and the last one is both are double

then we know that which one will be the maximum. So, we will see one more function, swap function using templates. So, swapping of two numbers. So, you are using the reference variable, all right. So, T address A which is equal to, right, what we are going to pass, right.



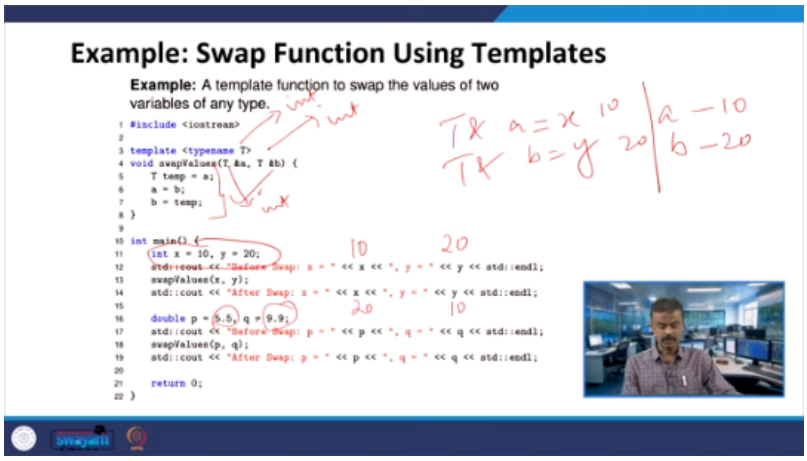
So, you are swapping two integers. So, which is a trivial one. So now quickly go to the main program x equal to 10 and y equal to 20 correct. So before swap we are printing x and y so 10 and 20. So before swap it will print x is 10 y is 20.

So now what you are doing is you are calling the swap values function, right? So passing x and y, so the address of a is equal to x and the address of b is equal to y. So that means a and x are pointing to the same location. B and Y are pointing to the same location. That means you are passing the reference, right? Both are pointing to the same location, correct?

So, here the value is 10 and 20. So, what will happen? A will also get the value 10, and B will also get the value 20. X is 10 you are passing, and Y is 20. So, A and B will have 10 and 20.

So, that means when I am passing an integer, a and b are integers, so automatically, since this is becoming an integer, this is also becoming an integer. The type name is integer, so that means you replace all t by integer and then swap, right? So when I swap, you know this is the logic. And here you get 20, and here you get 10, right? It is a call by reference or pass by reference. And the next one is when I am passing 5.5 and 9.9, right?

So, here it is becoming T address A equal to P and T address B equal to Q. Now, P and Q, right? Address of P and address of A both are same. Address of Q and address of B both are same. And here you have double. So the pointer is pointing.



The pointer location is double. Right. See how it is changing. The previous one we use T of address A equal to X. X is an integer. Right.

Both are pointing to the same location. Now A is pointing to the location of P. Right. So obviously with this syntax P and Q values will be assigned to A and B respectively. So, that means A will take 5.5 and B will take 9.9 and when you swap it right you are calling this swap algorithm will work for double everything will become double now right because your P Q are double A B will be automatically double. So, your T is becoming double I will just cross this integer because it was the last case and now T is becoming double correct.

So, once T is becoming double you are all T right in line number 4 obviously that was double and here T temp right. So, temp is becoming double A and B we have already seen they are double correct. So, quickly it is changing to double previously it was integer right. So, now it is a swapping of A and B

right where a is 5.5 and b is 9.9 and after swapping the p is becoming 9.9 p is becoming 9.9 and q is becoming 5.5 so previously it was 5.5 and 9.9 so these are the output you will get 10 20 is becoming 2010 5.5 9.9 is becoming 9.9 and 5.5 so if you run the code We go to the swap code. All right. So, the same explanation. So, program is running successfully.

You can see that. So, before swapping it is 10 and 20 and after swap it is becoming 20 and 10. All right. And before swapping it is 5.5 and 9.9. After swapping it is 9.9 and 5.5.

All right. So, I hope it is clear. Right. So, this is how it works. So, I have a swap values function.

So, it is a template function. So, it is swapping two variables. The variables should be of the same data type, right. So, that is what we have written. If you use only one T, the previous program had T1 and T2.

Accordingly, you can take care. So, when you use only T, that means type name T, so there should be only one variable, all right. And also, it works like a call by reference. And we have seen, So, this is the one we used: address a equals x means both address a and address x are the same, and address b and address y are the same.

And quickly, it is changing when it comes to line number 16. Now, the address of 'a' is pointing to the address of 'b', and the address of 'b' is pointing to the address of 'q'. I mean to say both are the same address, so obviously the value

will also be the same. So, that is what we had seen. In the function template, all right. So, in the next class, we will see some of the programs on functions and classes, right? And I will continue with the template function. So, then we will talk about the concept of generics also, right.

So, with this, I am concluding. Thank you so much.