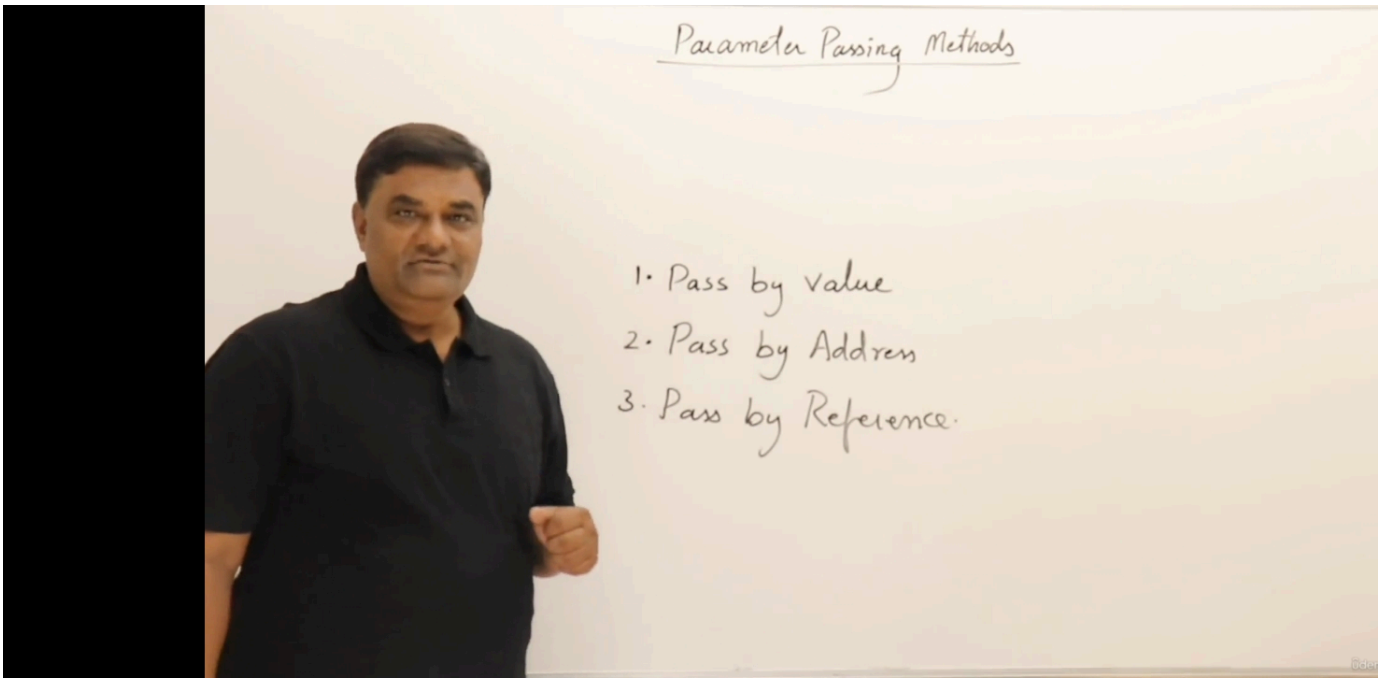
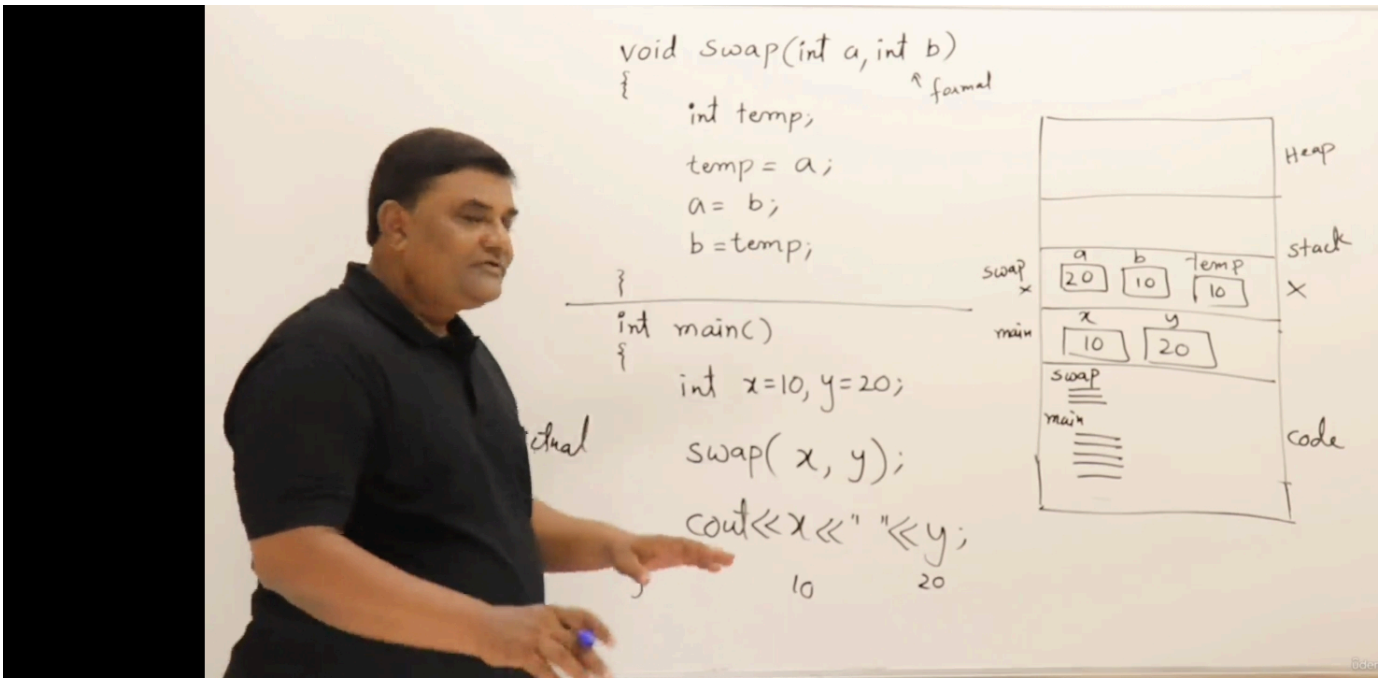

Parameter Passing Methods
1. Pass by value
2. Pass by Address
3. Pass by Reference.


void swap(int a, int b)
{
int temp;
temp = a;
a = b;
b = temp;
}
int main()
{
int x=10, y=20;
swap(x, y);
cout<<x<<" "<<y;
}
formal
actual
10
20
Heap
stack
code
swap
main
a
b
temp
20
10
10
x
y
10
20
x
swap
main

Parameter Passing Methods

Three parameter passing methods are supported by C++

Pass-By-Value : values of Actual parameters are passed to formal parameters. Actual parameters cannot be modified by function

Pass-By-Address: Address of Actual Parameters are passed to a function, formal parameters must be pointers. Function can indirectly access actual parameters.

Pass-By-Reference: Actual parameters are passed as reference to formal parameters, function can modify actual parameters.

Program for Call by Value

- Value of actual parameters are copied in formal parameters
- If any changes done to formal parameters in function, they will not modify actual parameters

```
Void swap(int a, int b)
{
        int temp;
        temp=a;
        a=b;
        b=temp;
}

Int main()
{
        int x=10, y=20;

        swap(x,y);
        cout<<x<<y;
}
```

Parameter Passing - FAQ

- **How Call by Reference works?**

In call by reference, complier **may** make a function as inline. The machine code of the function **may** be copied at the place of function call.

Or

Compiler **may** convert reference into a constant pointer.

(constant pointer: a pointer is initialised once and cannot be changed)

- **What happens, If one parameter is reference and another pointer?**

Obviously, function will not become an inline function. Compiler will convert a reference into constant pointer.