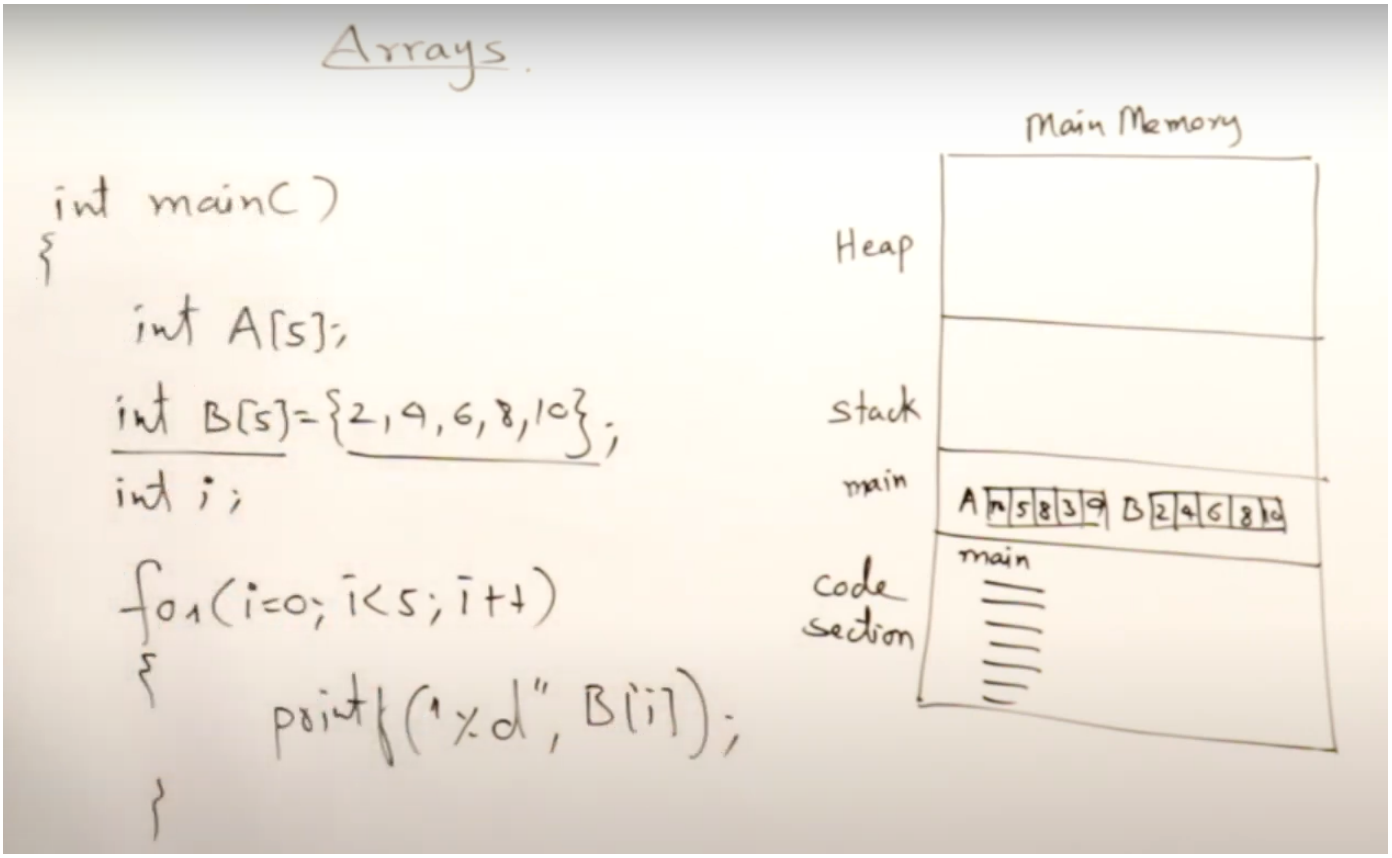
Abdul Bari DSA

- Array, Matrix, Linked List are used to store data.
- Stack, Queue, Heap, Tree, Graph, Hashing are there to use that stored data.
- Recursion and Sorting are ued to implement data structures.

Arrays -

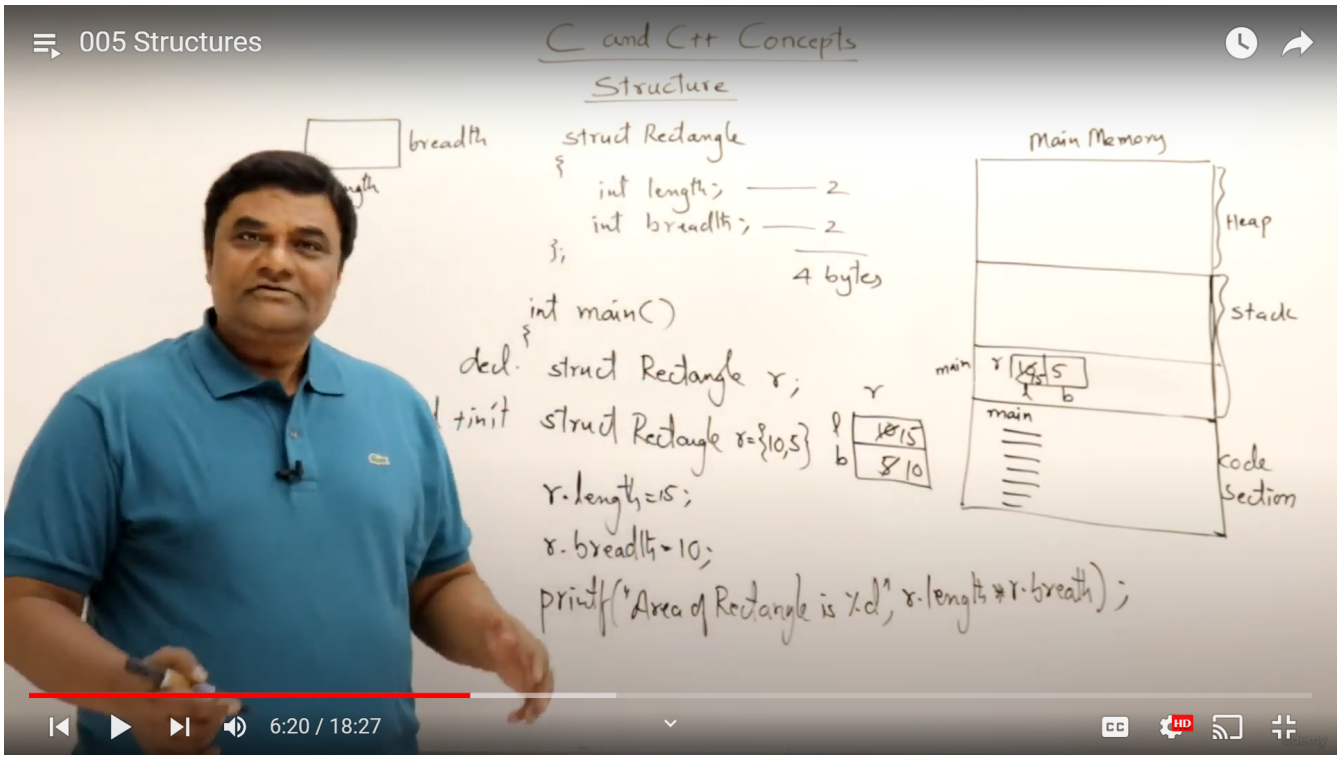


- Any Program/Function once starts executing , It data will appear in Stack(Including main function's), where it is accessible to code section of memory.
- Above photo shows Declaration, Declartion and Initialization and Accessing the elements of an array.
- To initialize all elements of an array with 0, we can just initilize 1st element with 0 and rest element will automatically be assigned value 0 by compiler.

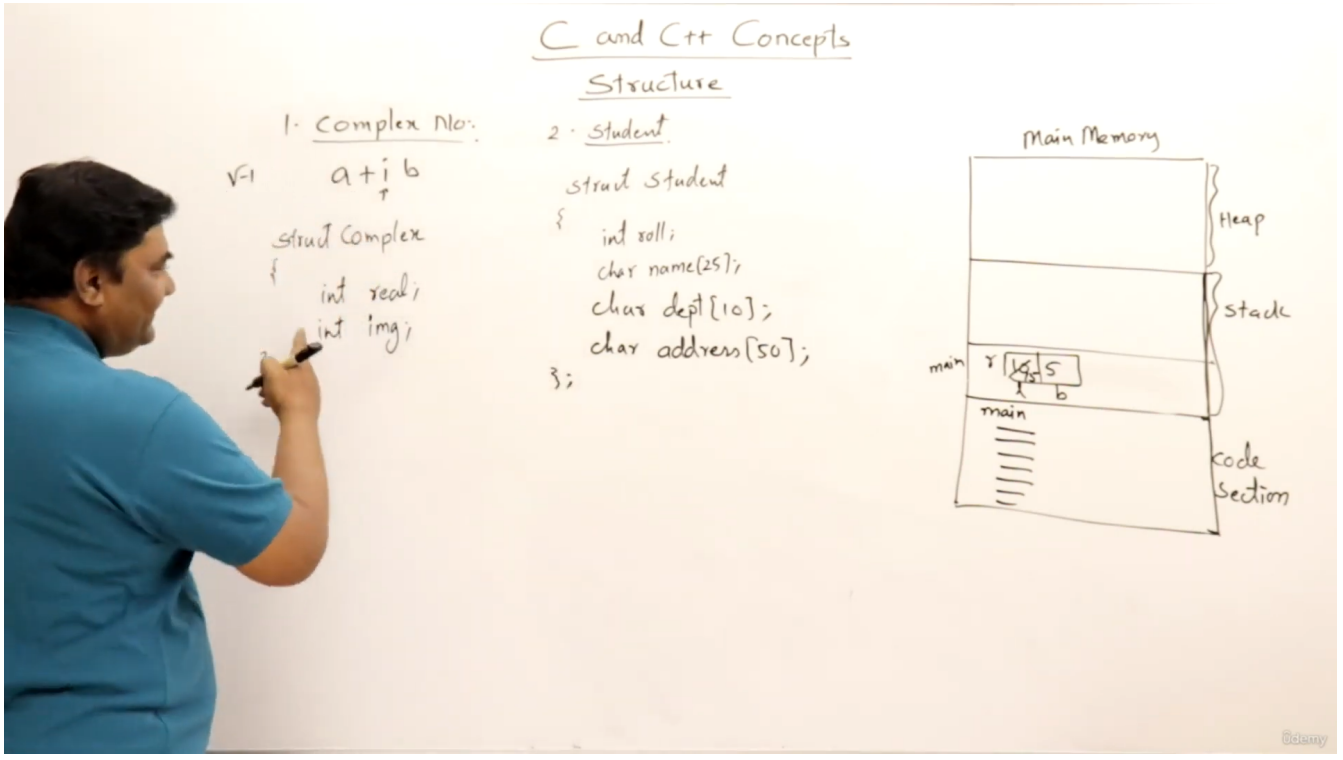
```
int A[10]={0};
```

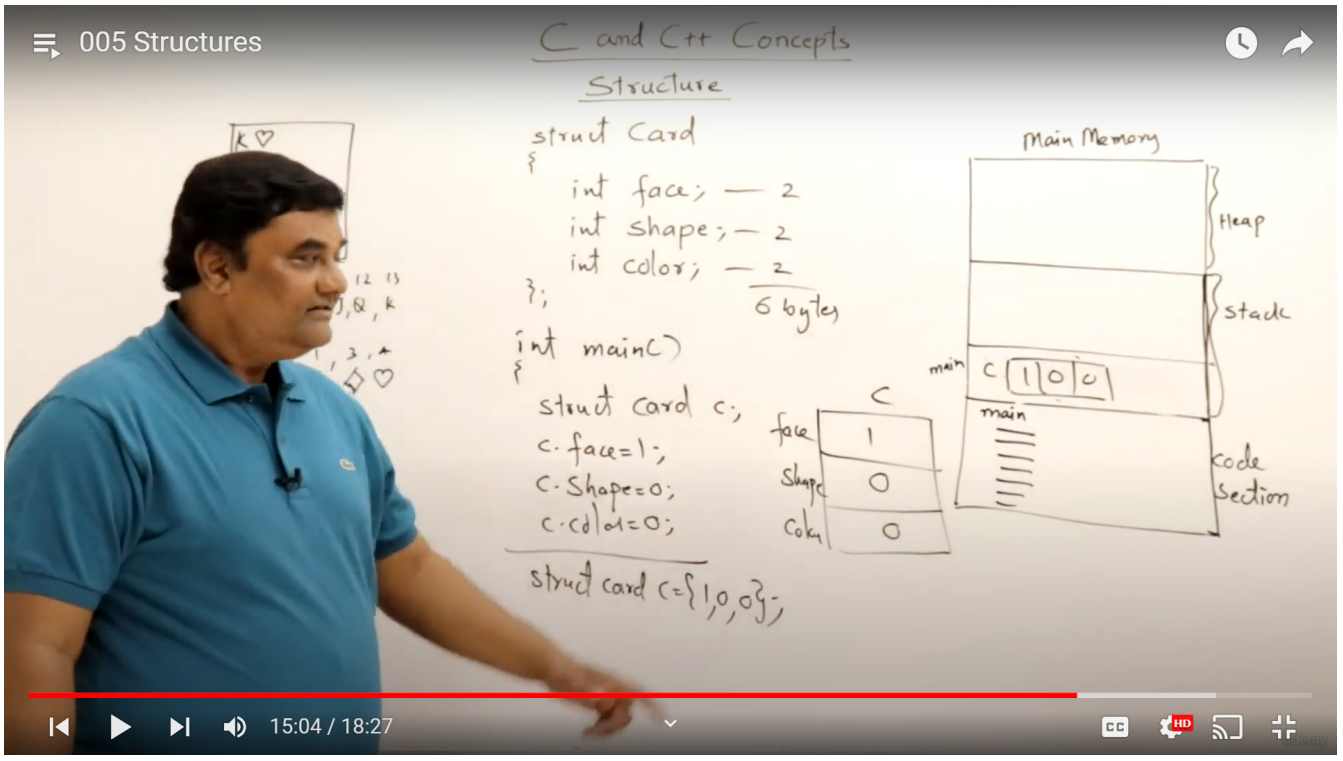
- If unitialized then, it will contain garbage value.
- Variable length array - ask user for size and then ask to give the elements to initialize.

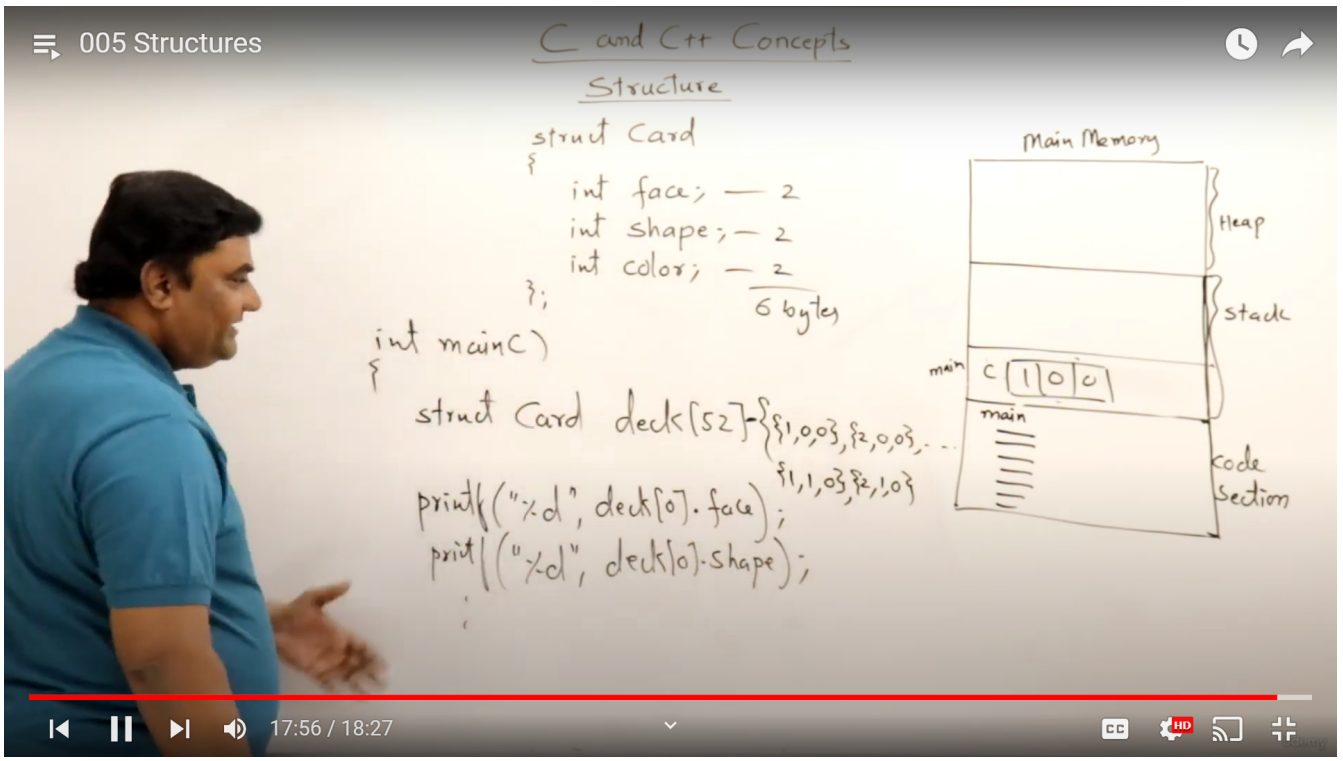
Structures -



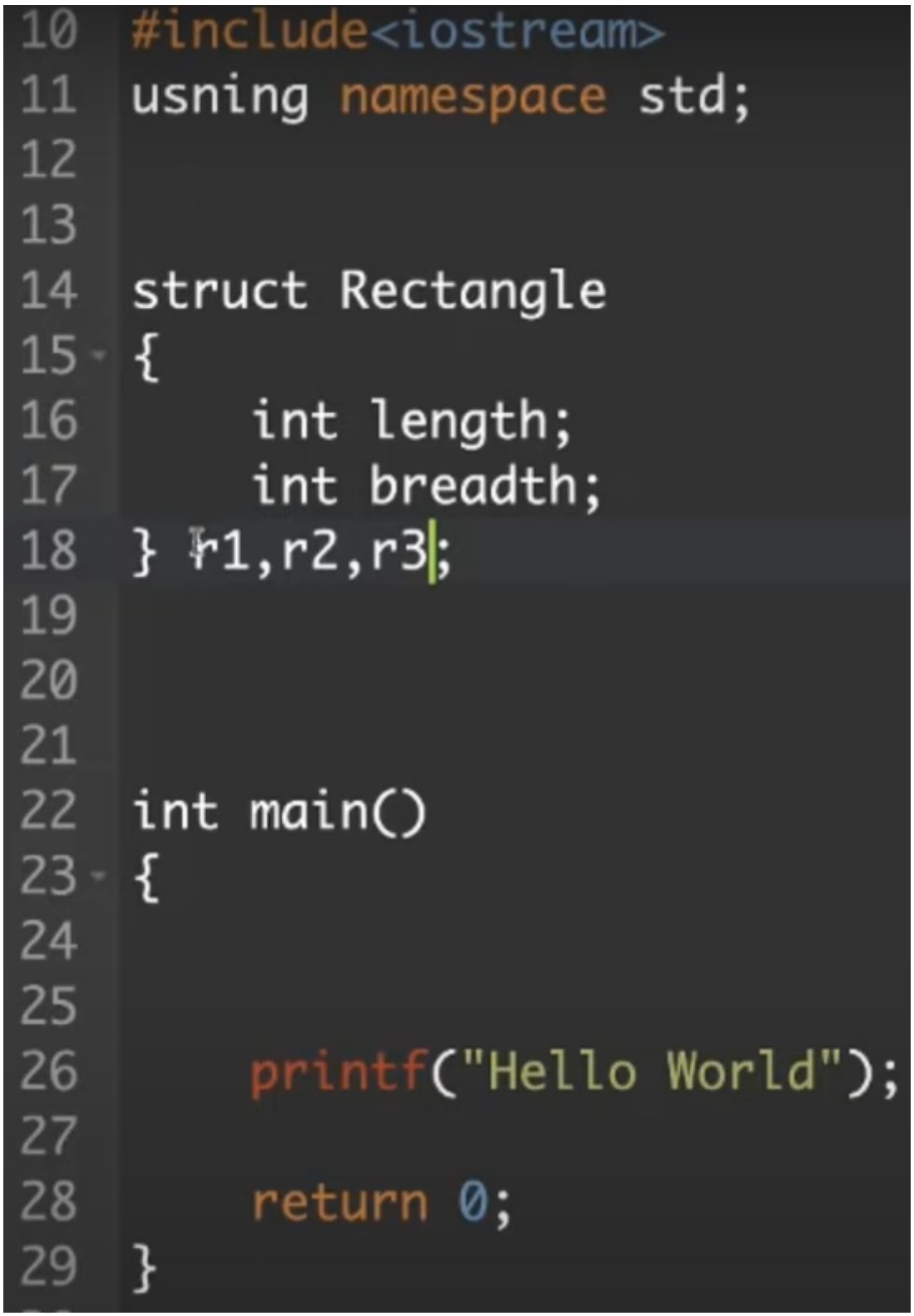
- Just writing definition of our structure won't allocate memory, once we create variable of that struct type then memory is allocated.
- Used to we want to declare an element having more than one values associated with it(that too of different data types usually).
- Examples of a structure -




005 Structures
C and C++ Concepts
Structure
struct Card
{
int face; — 2
int shape; — 2
int color; — 2
};
6 bytes
int main()
{
struct card c;
c.face=1;
c.Shape=0;
c.color=0;
struct card c={1,0,0};
c
face 1
Shape 0
Color 0
Main Memory
Heap
Stack
main c 1 0 0
main
Code Section
15:04 / 18:27


005 Structures
C and C++ Concepts
Structure
struct Card
{
int face; — 2
int shape; — 2
int color; — 2
};
6 bytes
int main()
{
struct Card deck[52]={{1,0,0},{2,0,0},...
{1,1,0},{2,1,0}
printf("%d", deck[0].face);
printf("%d", deck[0].shape);
Main Memory
Heap
Stack
main c 1 0 0
main
Code Section
17:56 / 18:27

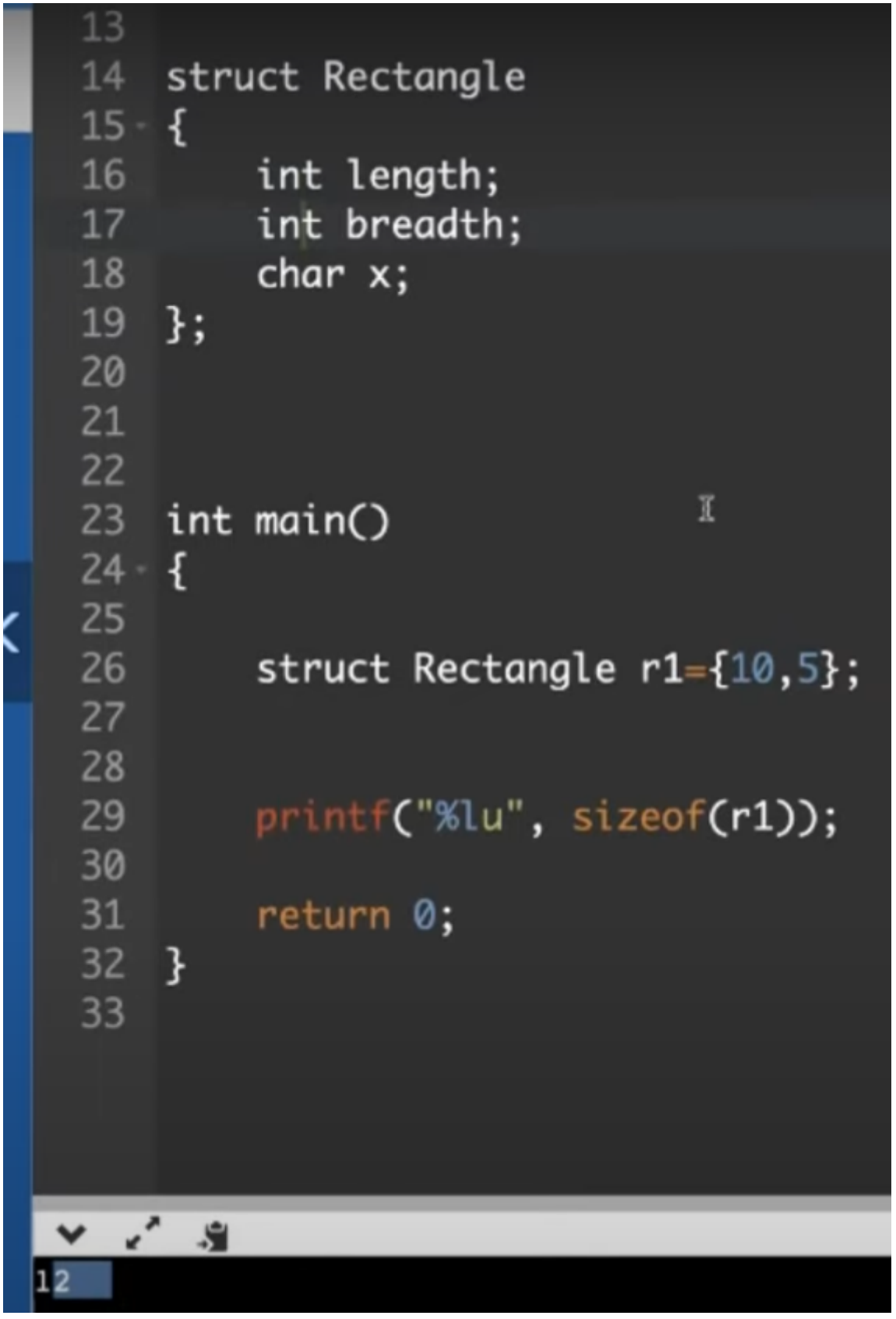
- Definition along with declaration of structures -



```
#include<iostream>
usning namespace std;


struct Rectangle
{
    int length;
    int breadth;
} r1,r2,r3;



int main()
{

    printf("Hello World");

    return 0;
}
```

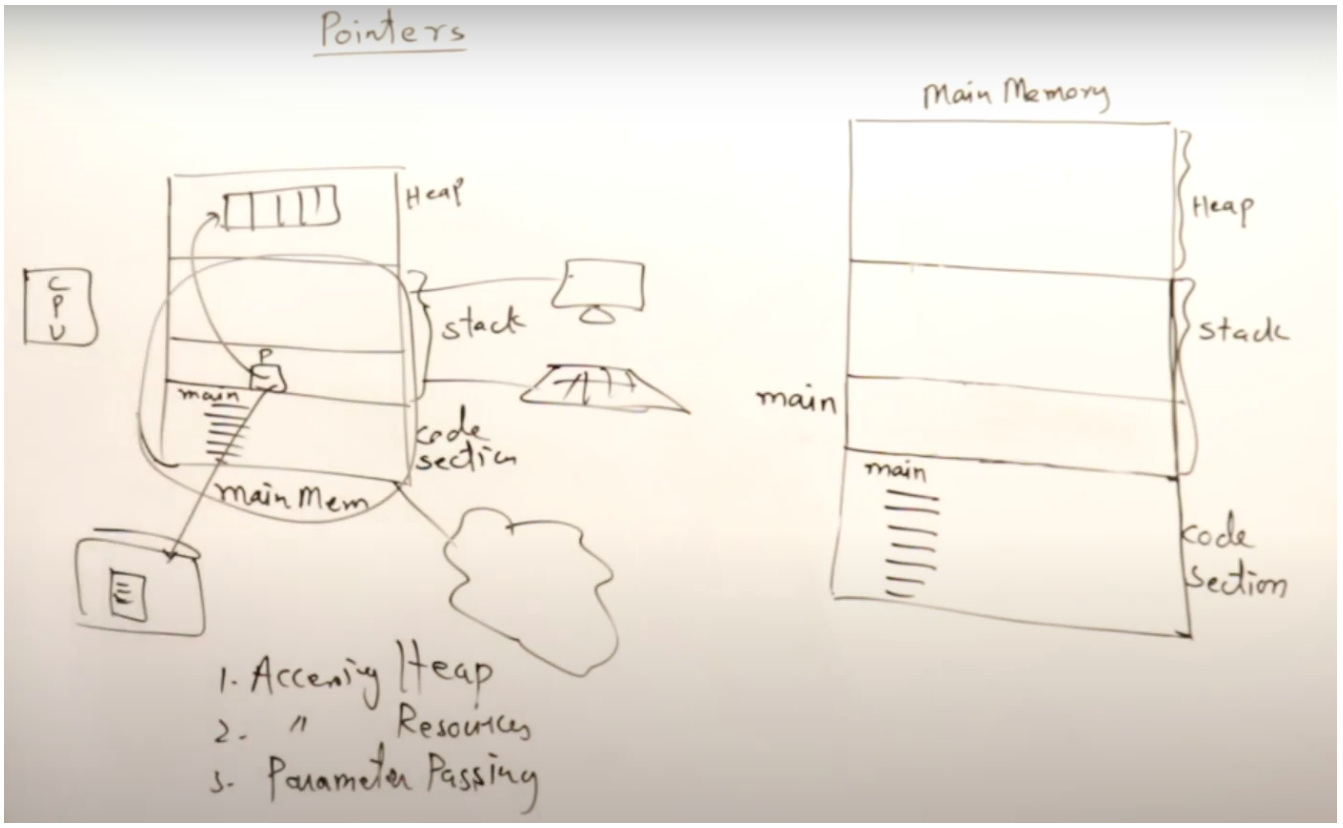
- Padding in C -
  If char is inside structure, its assigned 4 bytes instead of 1 byte , b/c machine reads 4 bytes in one

go

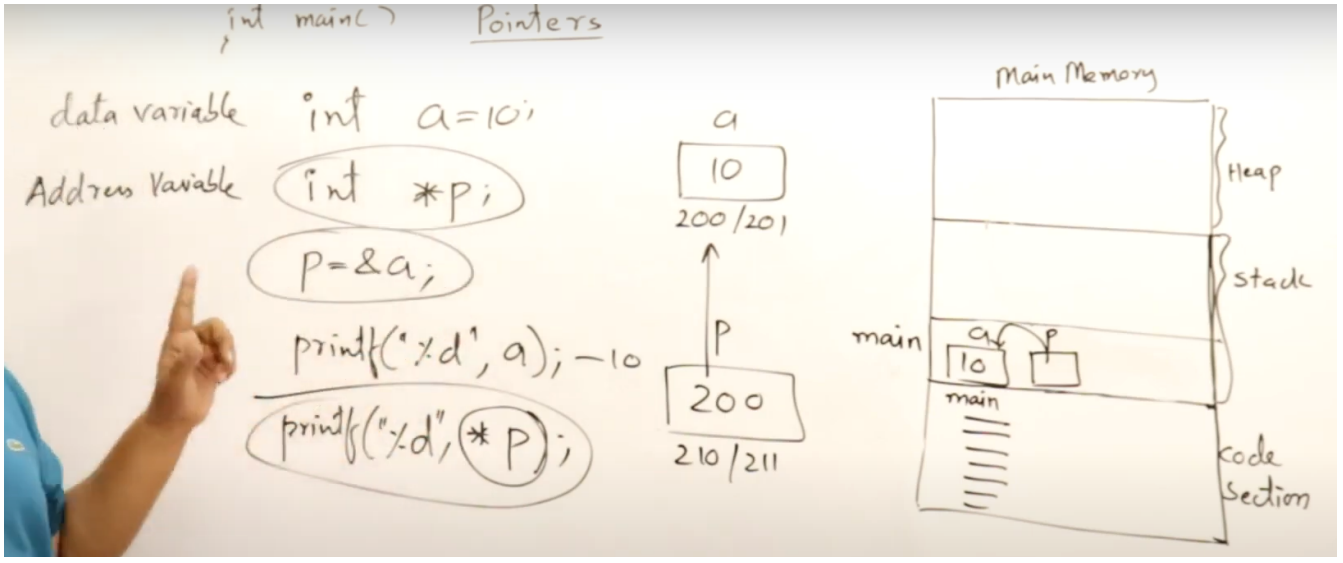


Pointers -

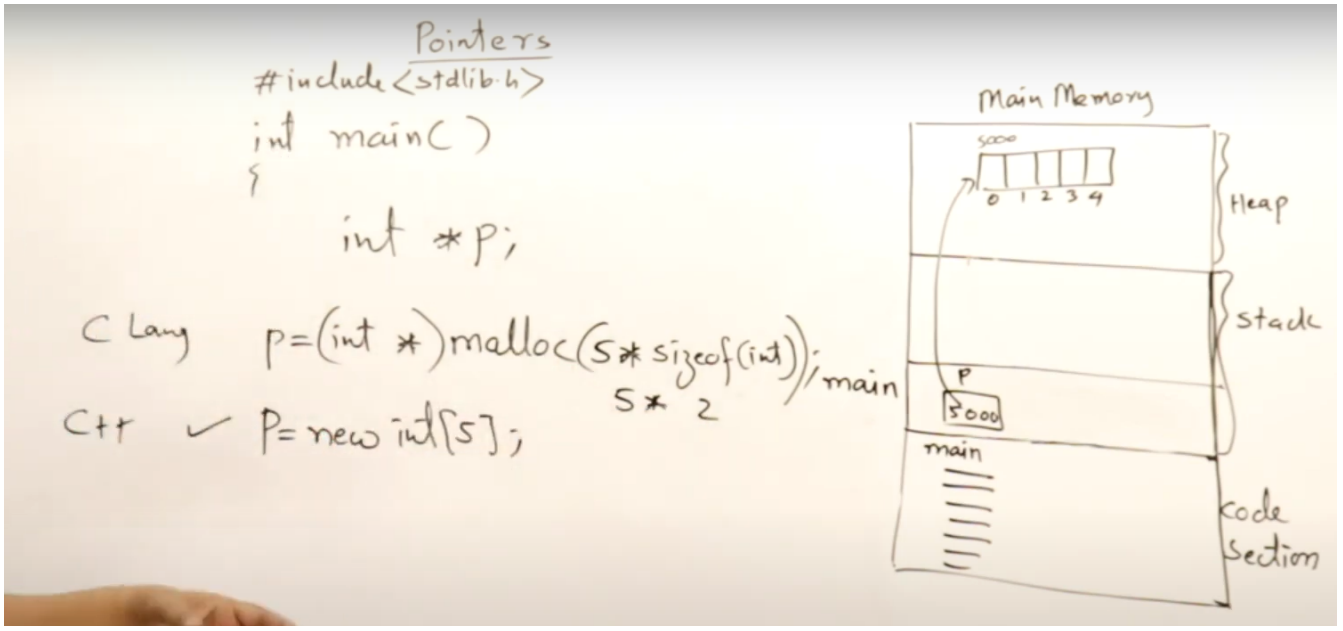
- address storing variables are called as Pointers.
- Pointers are used to access memory which is not available to program directly.
- Stack memory is available to program directly, but heap memory is not accessible directly. Similarly, resources like keyboard, mouse, filesystem ..etc are not accessible directly.



- To access heap, indirect resources , we need pointers b/c pointer variable is inside stack only, but it can point to indirect resources.
- Pointers are also used to Pass Parameter



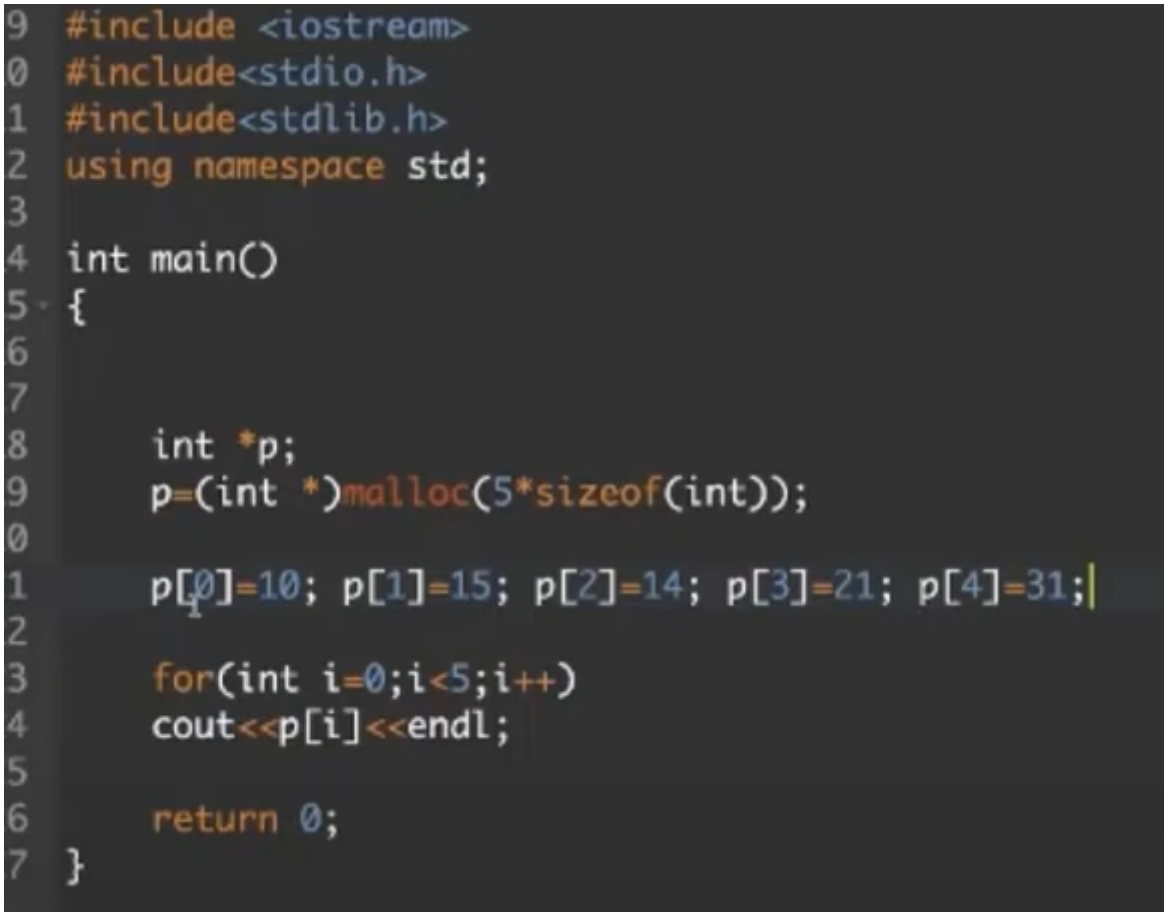
- How to use heap memory (using pointers)-



- By default, malloc returns a void pointer , we have to typecast it to use it. Note - malloc is in stdlib.h library
- We need to use star to declare and dereference/use pointers,
  and no need of star to assign and do pointer arithmetic.
- An array name acts like a pointer to its first element
  so to assign an array to a pointer, no need of "&" operator.

```
10 #include<stdio.h>
11 using namespace std;
12
13 int main()
14 {
15
16     int A[5]={2,4,6,8,10};
17     int *p;
18     p=A;
19
20     for(int i=0;i<5;i++)
21     cout<<p[i]<<endl;
22
23     return 0;
24 }
```

- Array inside heap -

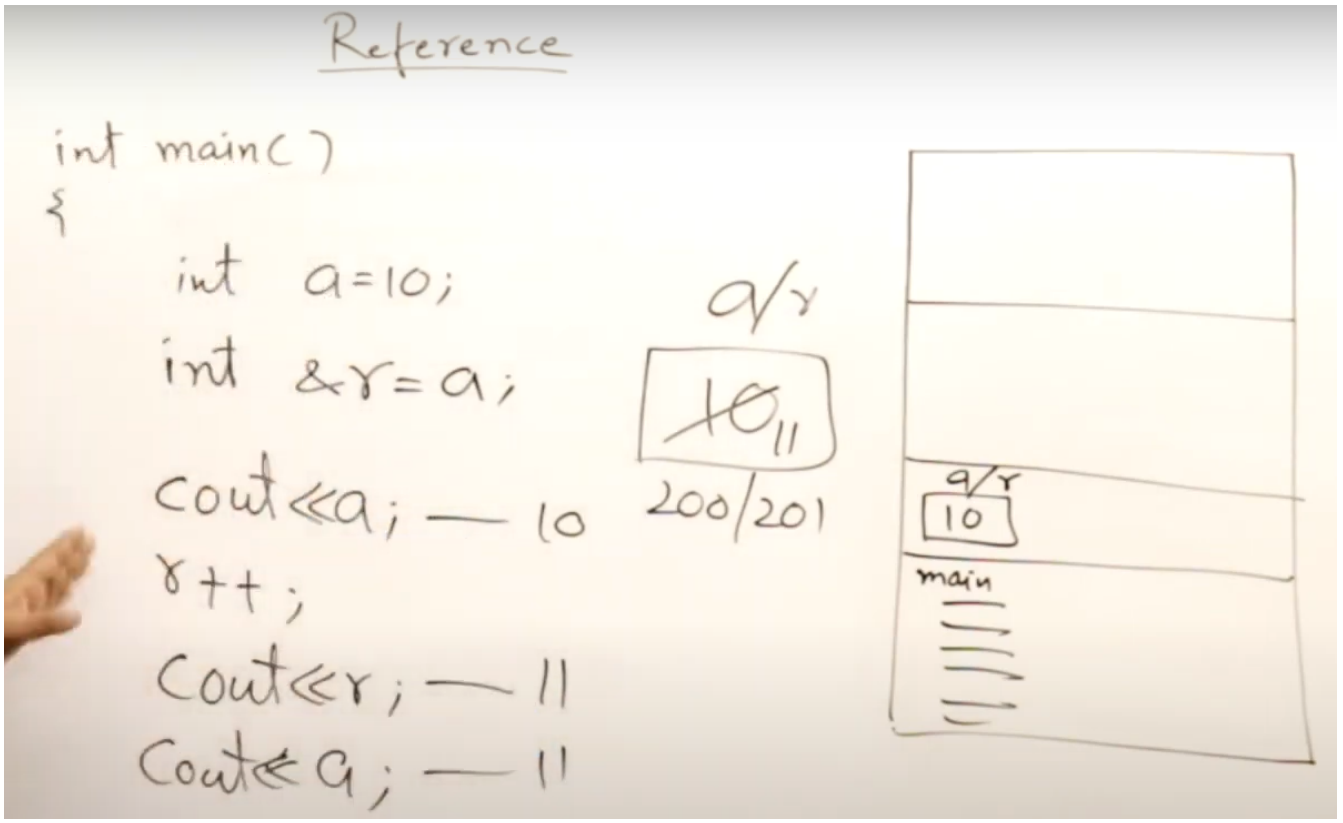


```
#include <iostream>
#include<stdio.h>
#include<stdlib.h>
using namespace std;

int main()
{


    int *p;
    p=(int *)malloc(5*sizeof(int));

    p[0]=10; p[1]=15; p[2]=14; p[3]=21; p[4]=31;

    for(int i=0;i<5;i++)
    cout<<p[i]<<endl;

    return 0;
}
```

- We must delete the heap memory after using it -
  using free(p); - in c
  using delete [ ] p; - in c++
- Once program ends, even heap memory is released
- Size of pointer - 8 bytes in 64bit machine(irrespective of datatypes it is pointing to)

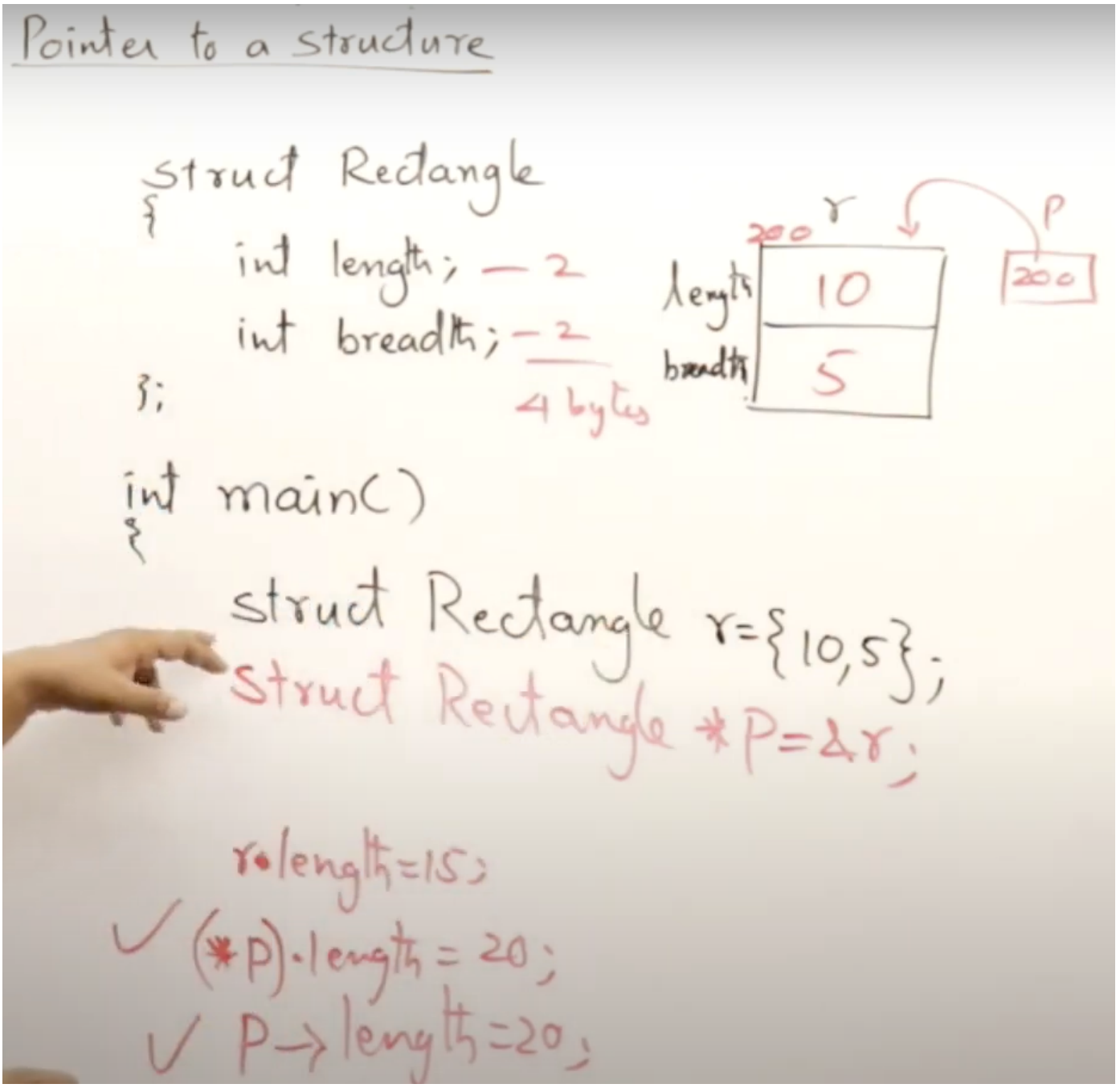
Reference(only in C++)-

- reference is just nickname/alias to any variable in c++.
- It is used in paramter passing.
- We have to declare and assign at the same time in case of reference.
  int &r = a;



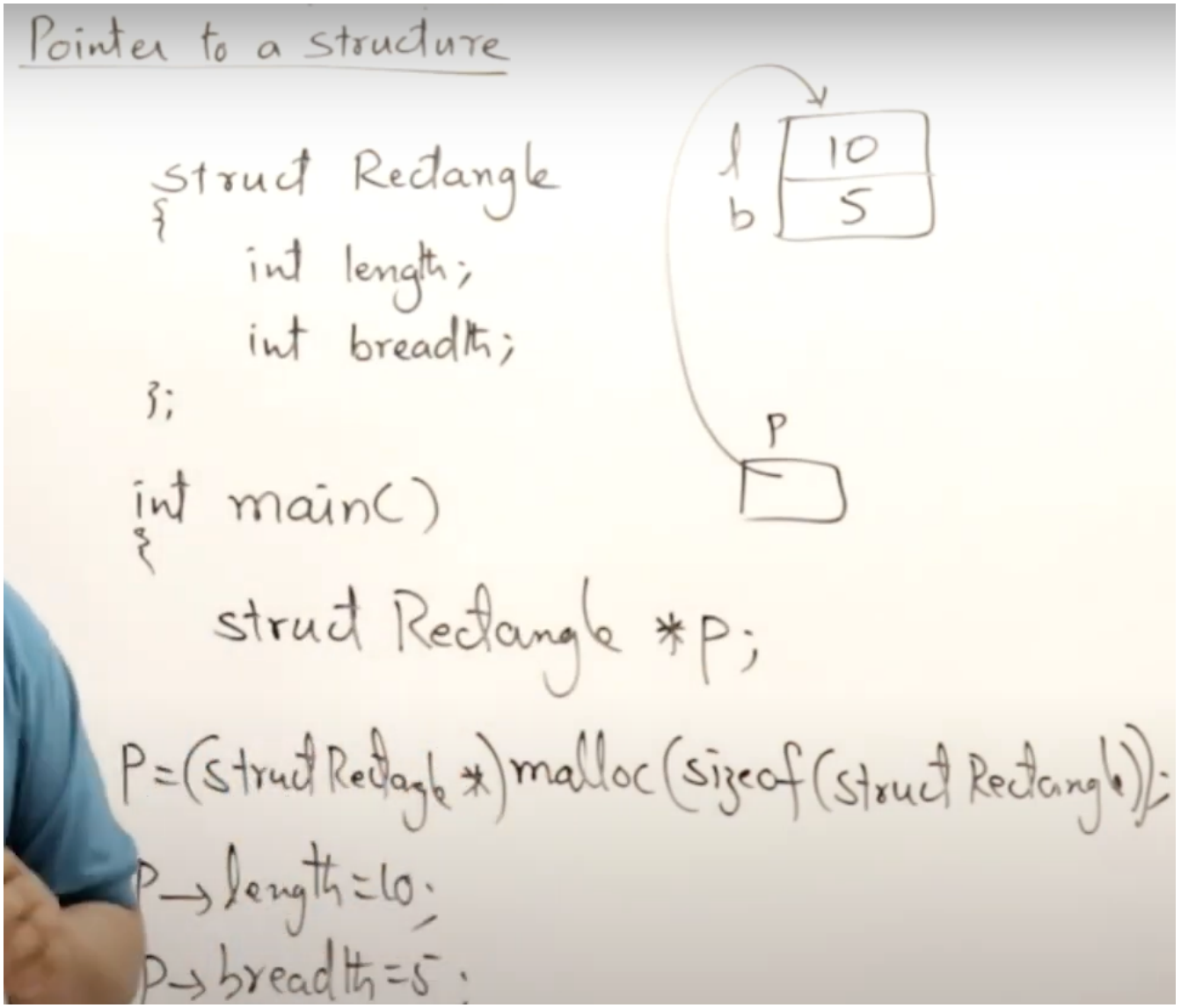
- Once defined a reference, it cannot be re-reference to any other variable.
- Reference doesn't take any memory.

Pointer to a Structure -



- use -> to access a structure using pointer

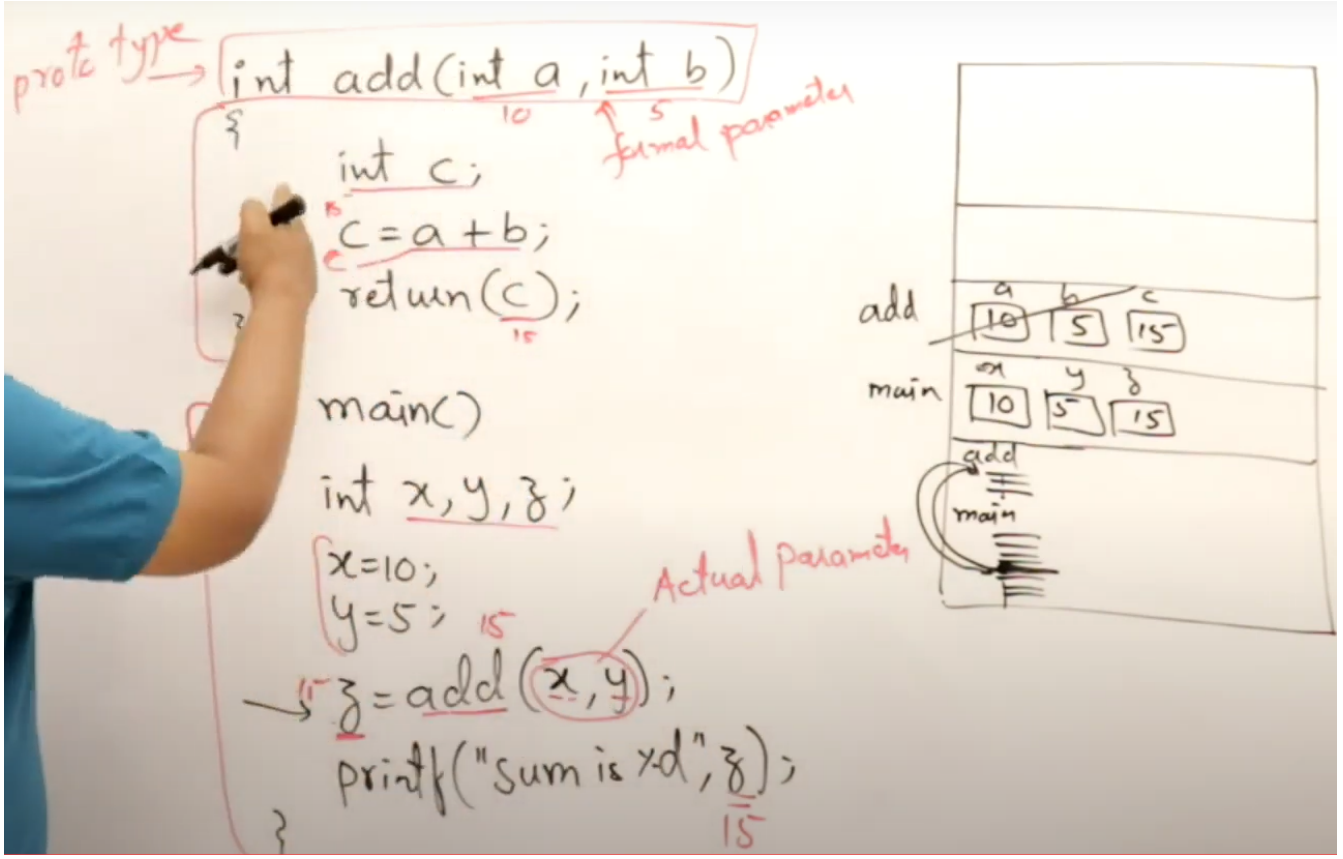
- Creating a structure in Heap -



- in C++ no need to write struct before structure names while initialization

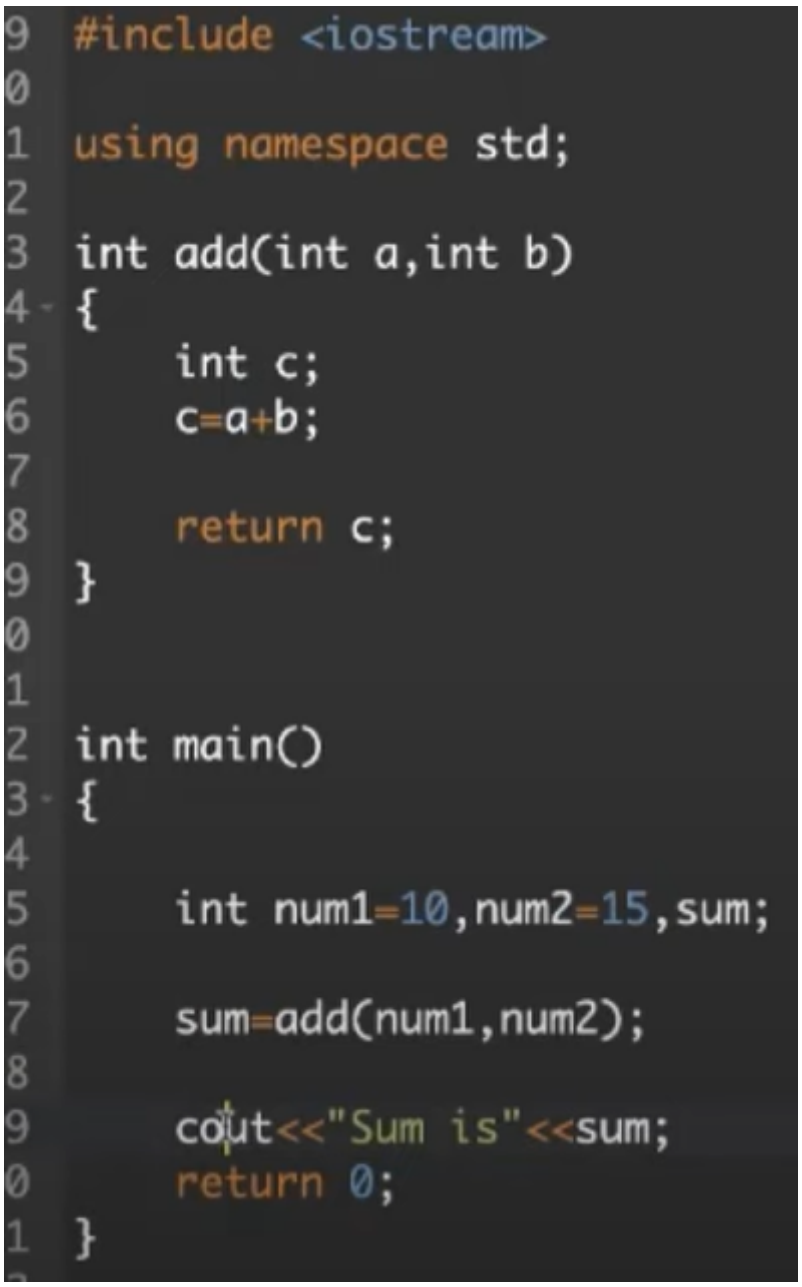
Funtions -

- It is a collection of similar Instructions to perform a specific Tasks.



- Any function can't access variables of another Function.
- Monolithic Programs are those in which everythingh is written inside main function inly.
  Modular Programs are those in which program is divided into several different functions called

Modules. This type of programming is called as Modular/Procedural Programming.



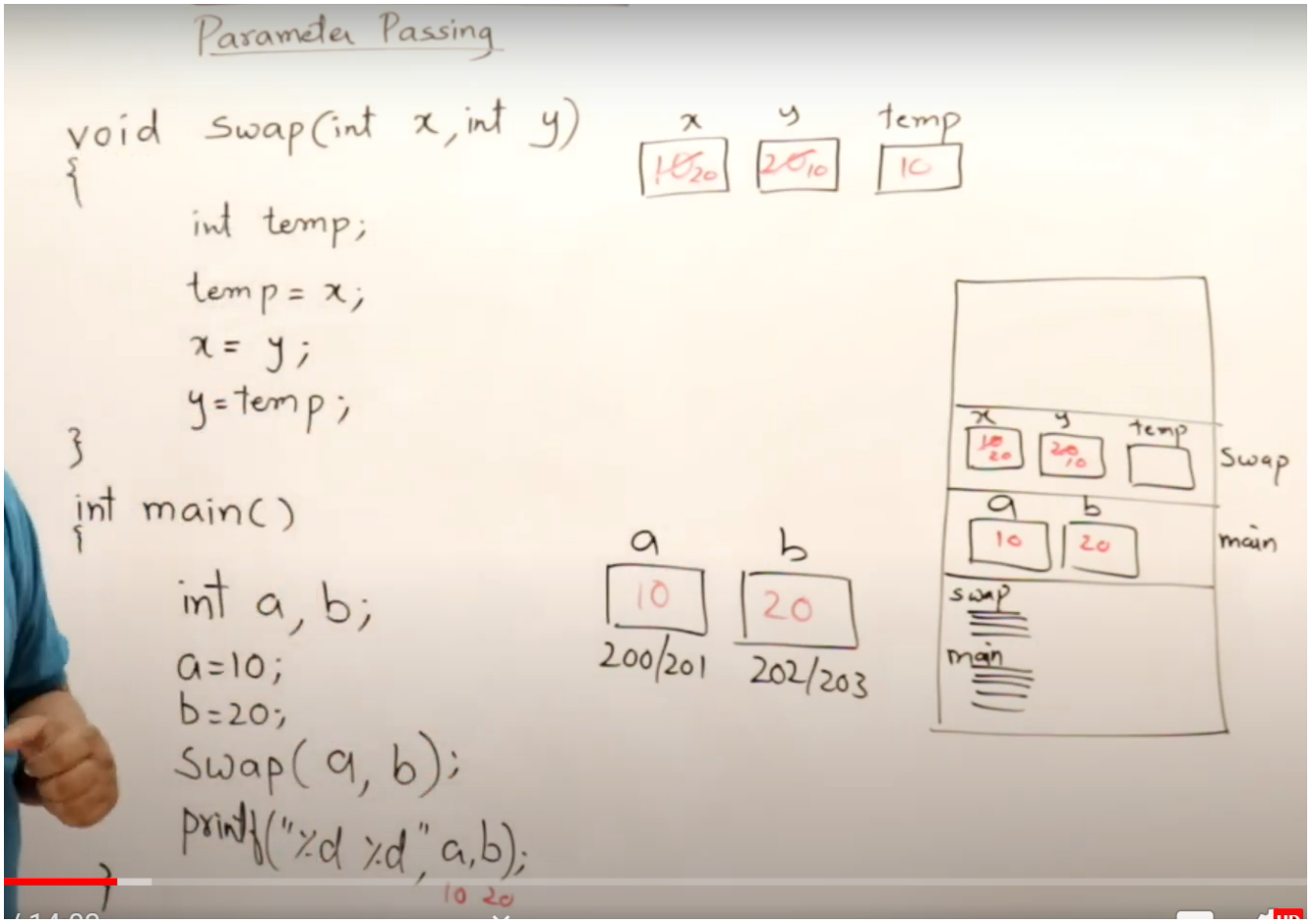
```
#include <iostream>

using namespace std;

int add(int a,int b)
{
    int c;
    c=a+b;

    return c;
}


int main()
{

    int num1=10,num2=15,sum;

    sum=add(num1,num2);

    cout<<"Sum is"<<sum;
    return 0;
}
```

Parameter Passing -

1. Call/Pass by Value -

- Formal Parameters are modified and no change to actual Parameters.
- Formal and Actual variables occupy different stacks.

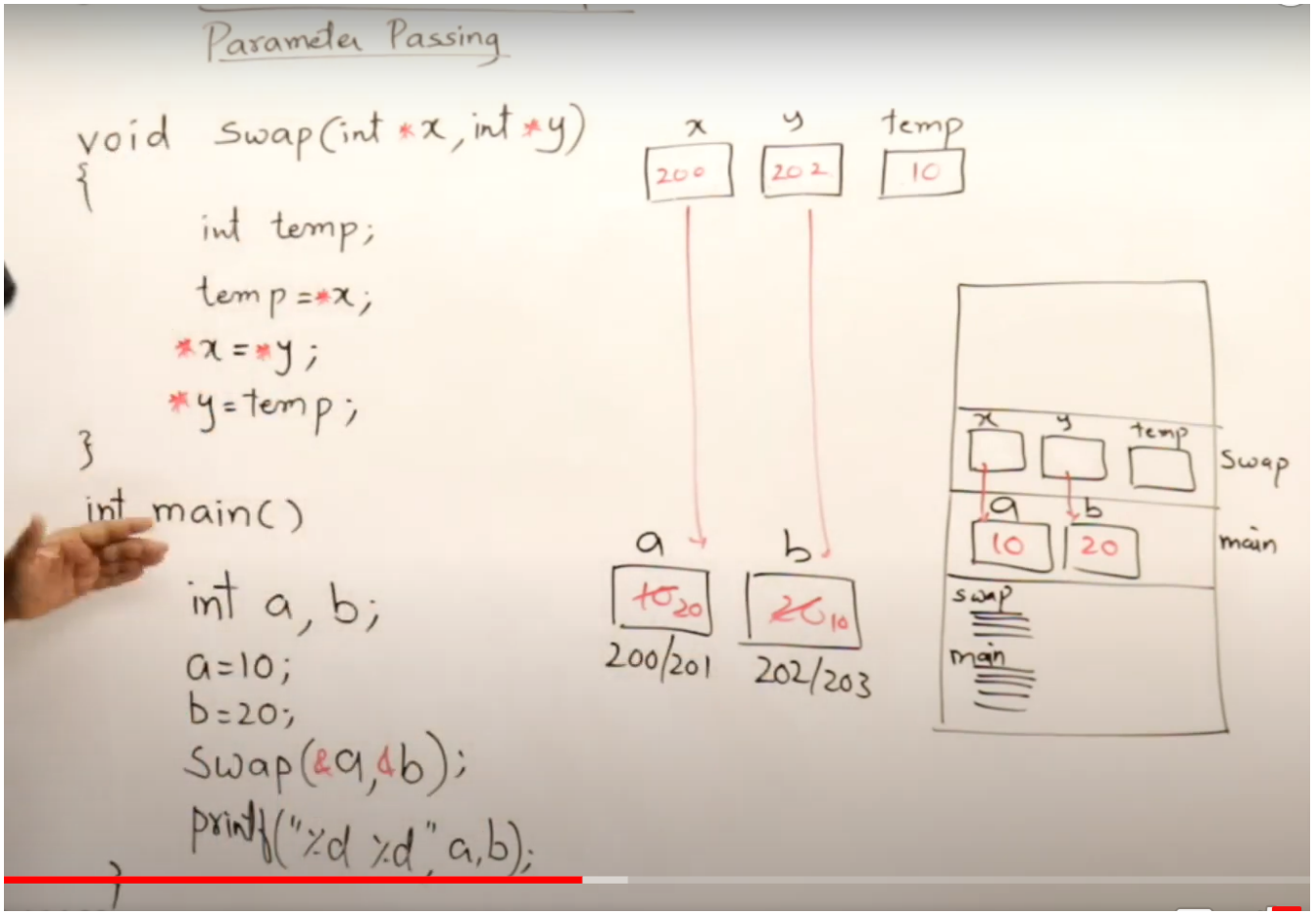
- Suitable to use when function return the results, and don't actually modifies the actual parameters.



2. Call/Pass by address -

- Formal variables are pointers to Actual variabkes
- Actual Parameters are modified using Pointers
  -Suitable to use when more than one parameters to be returned by a function or want to modify

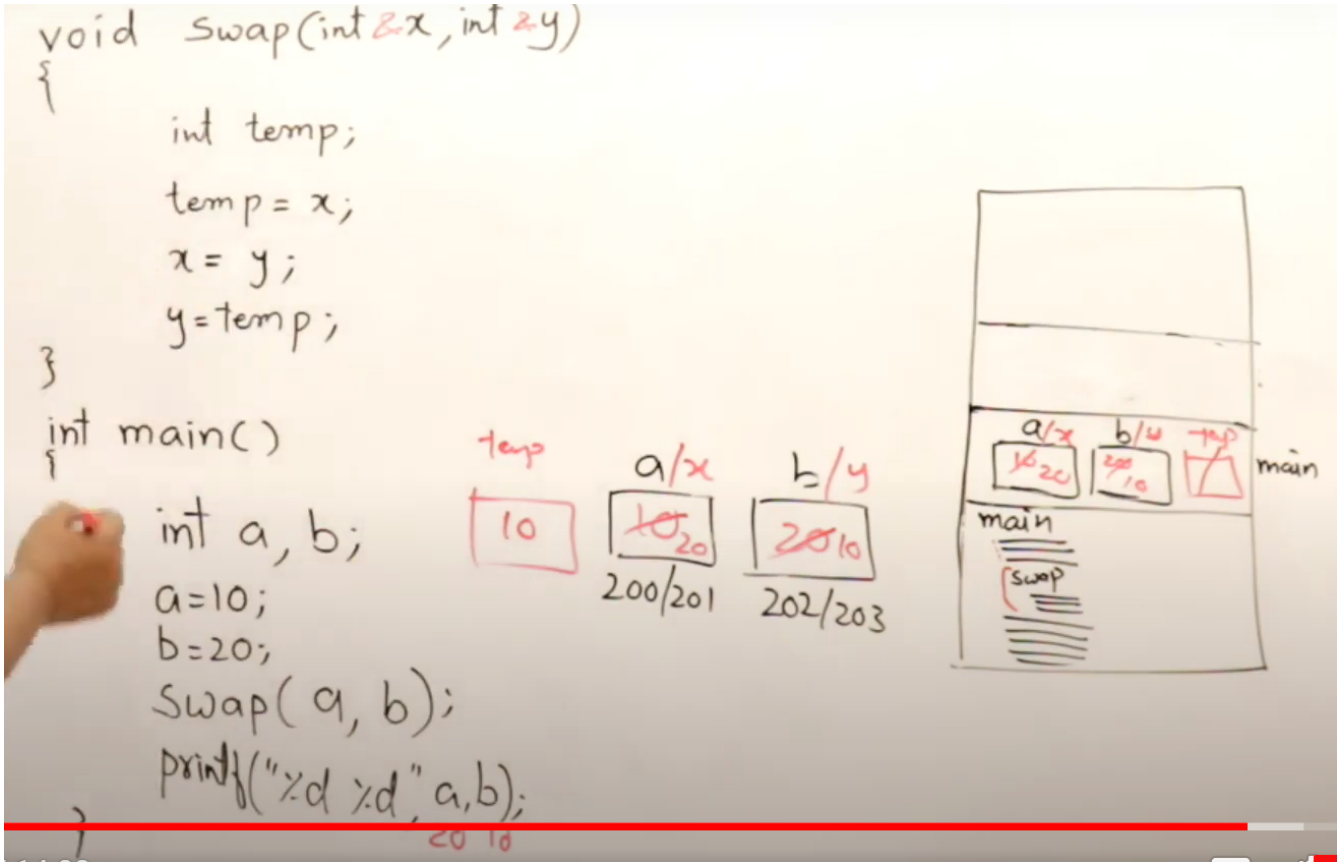
actual parameters.



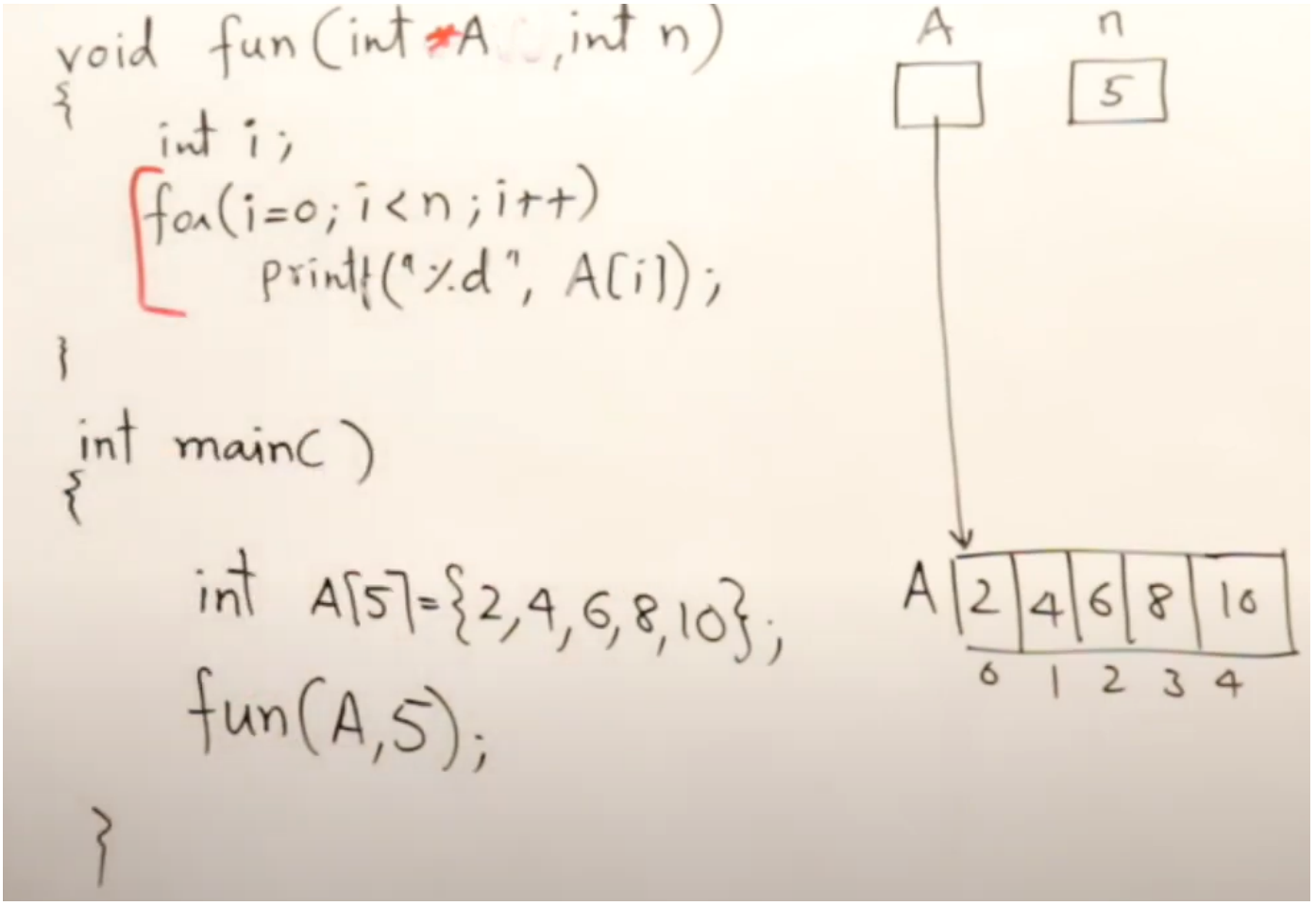
3. Call/Pass by Reference(only in c++)-

- Just another name given to actual Parameters.
- Since only alias , so No separate memory for formal Parameter in stack
- Code of function is copied in place of function call
- only use for small functions, without loops and all.

- These functions may or maynot be implemented as inline functions by compilers



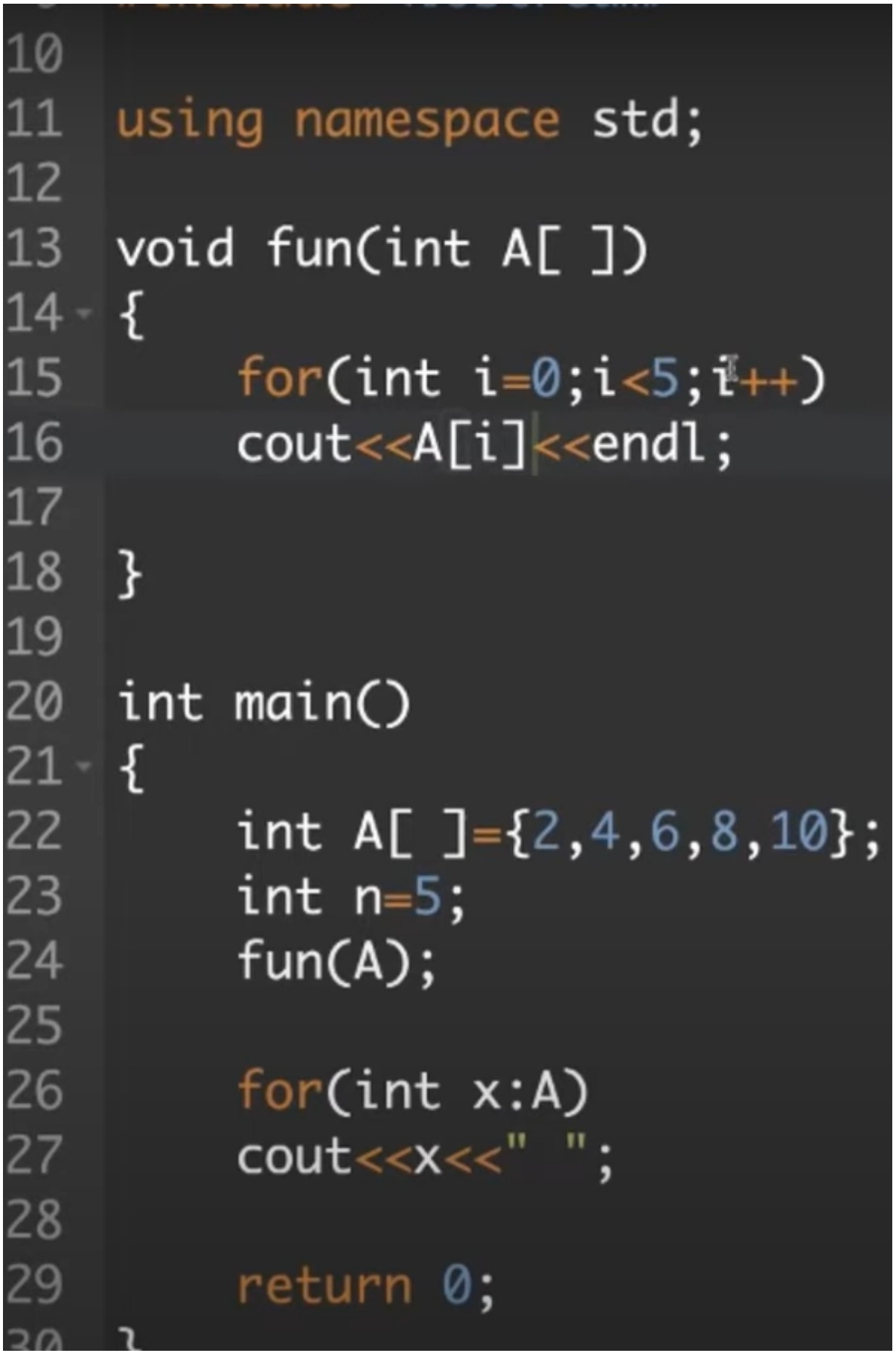
Array Passed as Parameter -



- We have to pass array name (which is a pointer to First element) and size of array(b/c if we use sizeof(A) inside to get size of array, it will return size of pointer A)
- To store pointer to 1st element to an array, we can use either `int A[] or int *A.`

```
int A[]   - can only point to an array
int *A   - can point to any integer, including Integer array.
```

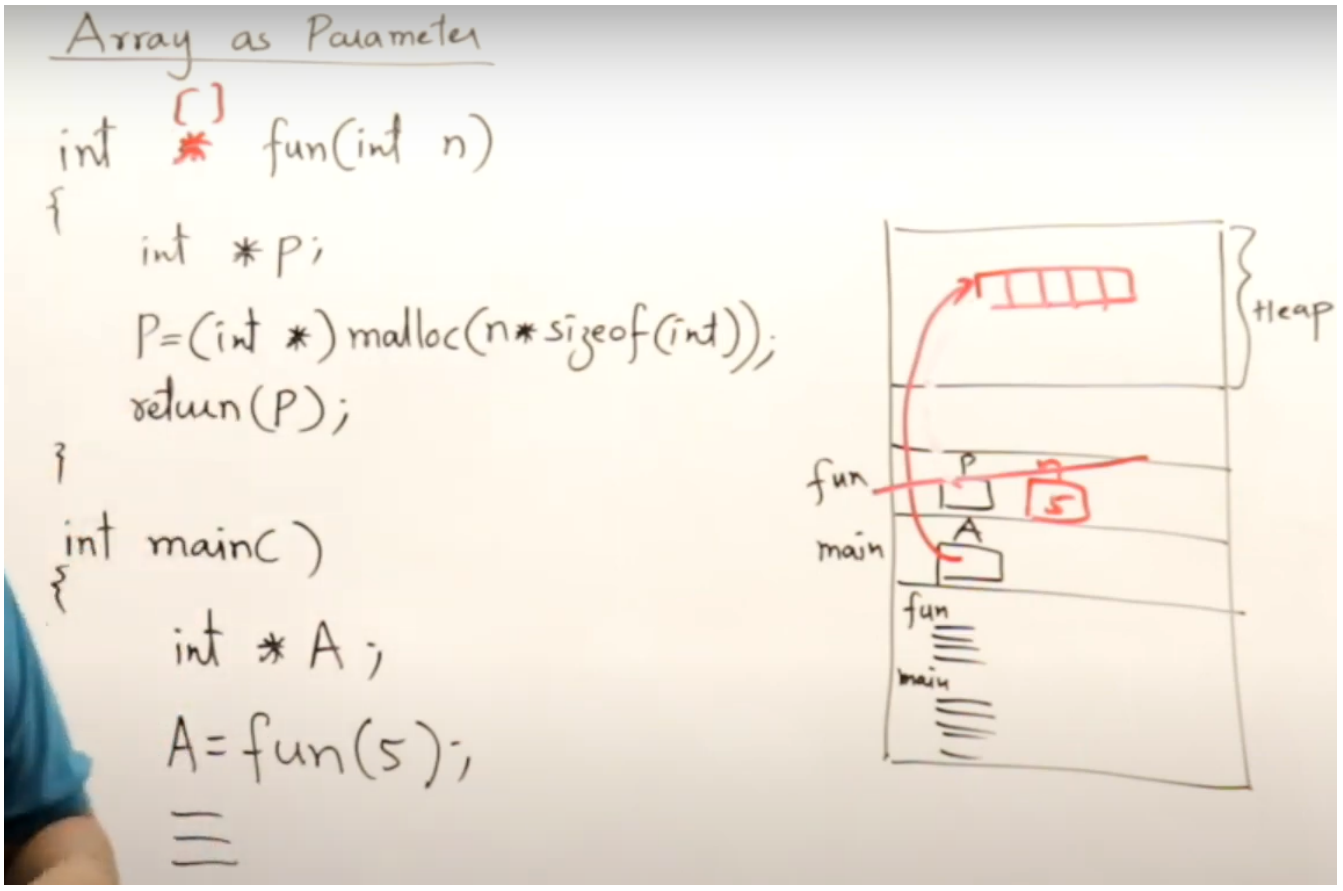
- If array passed as parameter, then we can't use for each loop in C++ to print elements of an array b/c for eac loop can't iterate over an pointer. Use for loop instead



```
10
11  using namespace std;
12
13  void fun(int A[ ])
14  {
15      for(int i=0;i<5;i++)
16      cout<<A[i]<<endl;
17
18  }
19
20  int main()
21  {
22      int A[ ]={2,4,6,8,10};
23      int n=5;
24      fun(A);
25
26      for(int x:A)
27      cout<<x<<" ";
28
29      return 0;
30  }
```

Array Returned as Parameter -

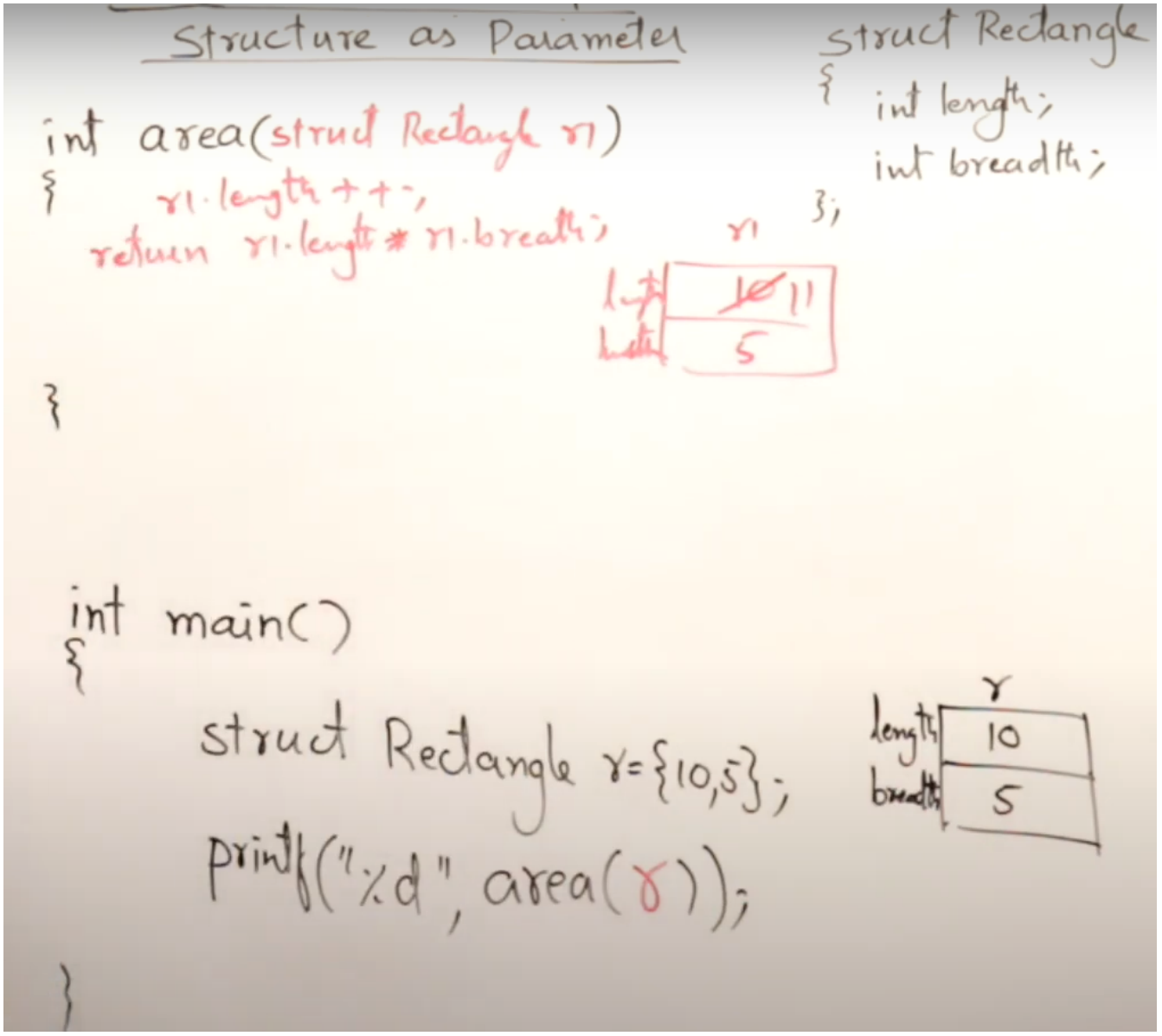
- return type should be `int [] or int *`



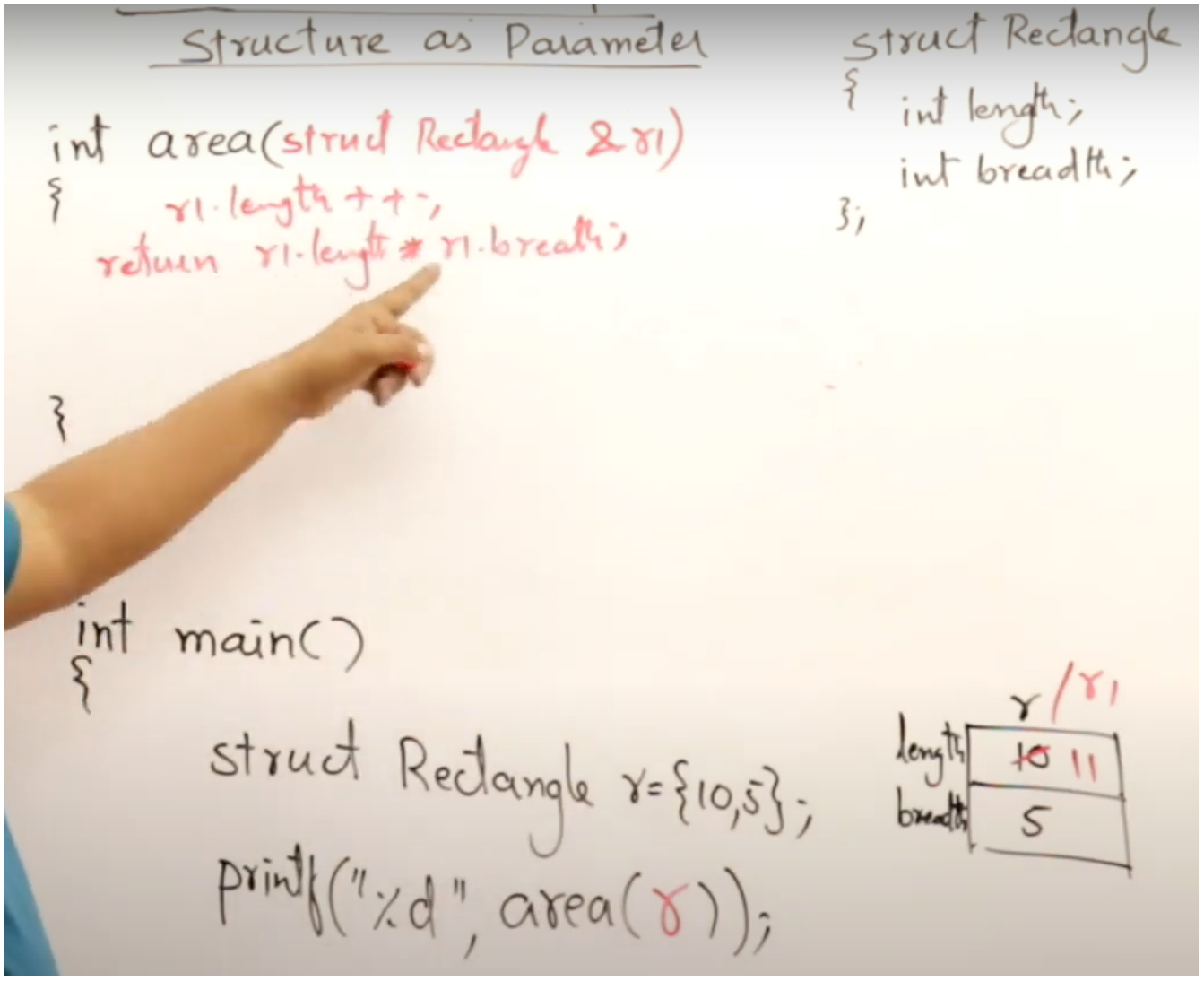
Structure Passed by Parameter -

- can be passd by value, reference or address

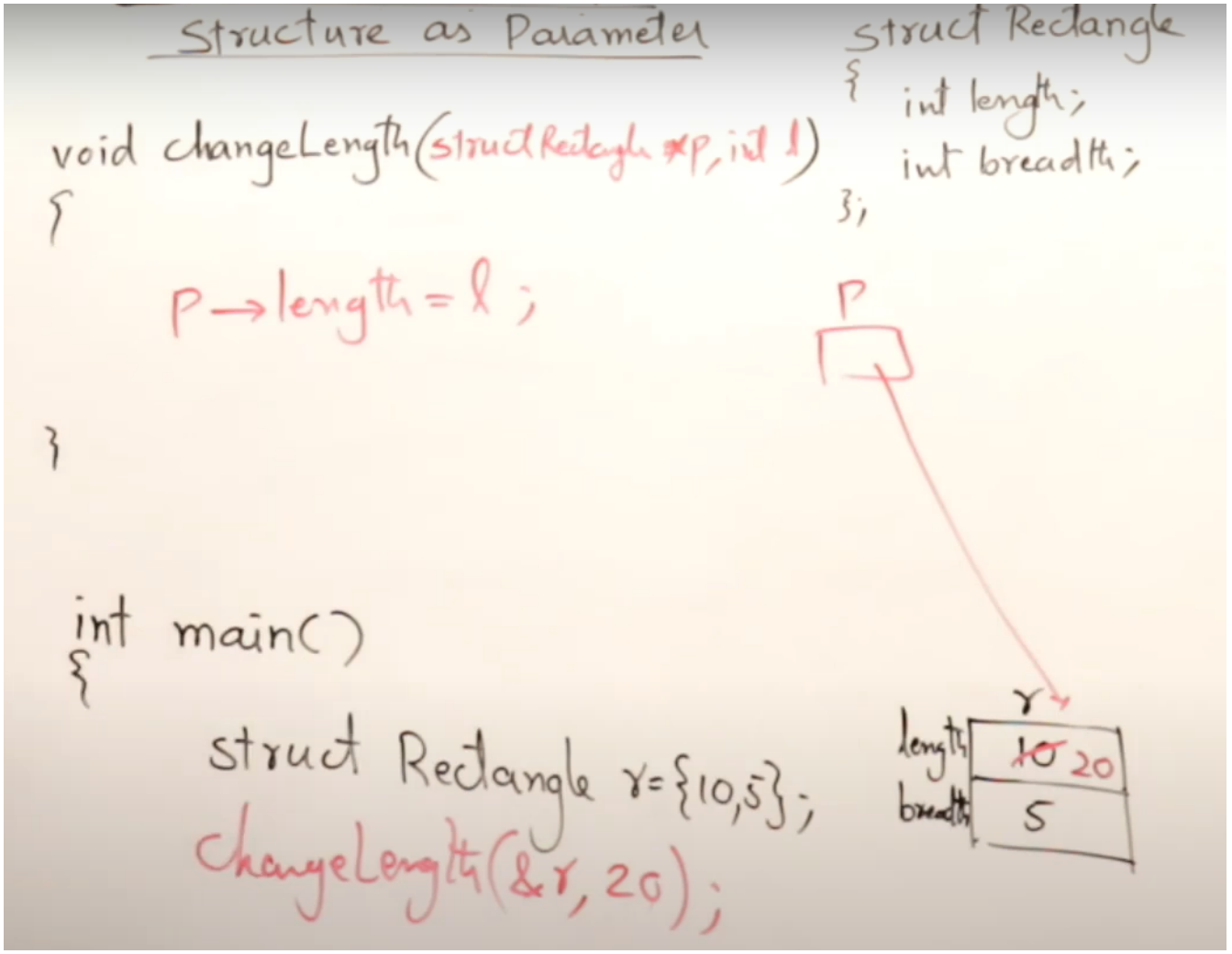
1. Pass by value -



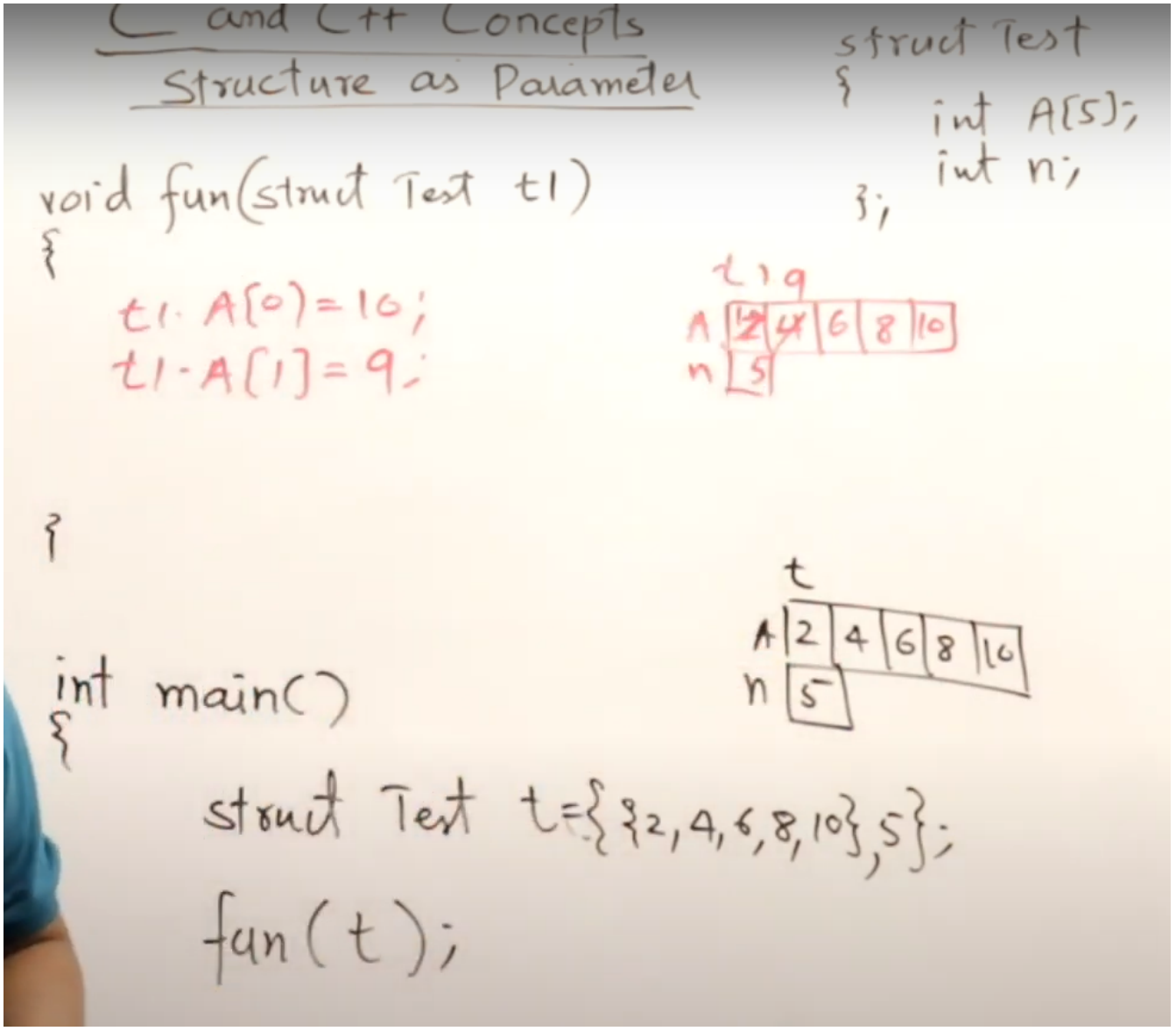
2. Pass by reference(only in c++) -



3. Pass by address -



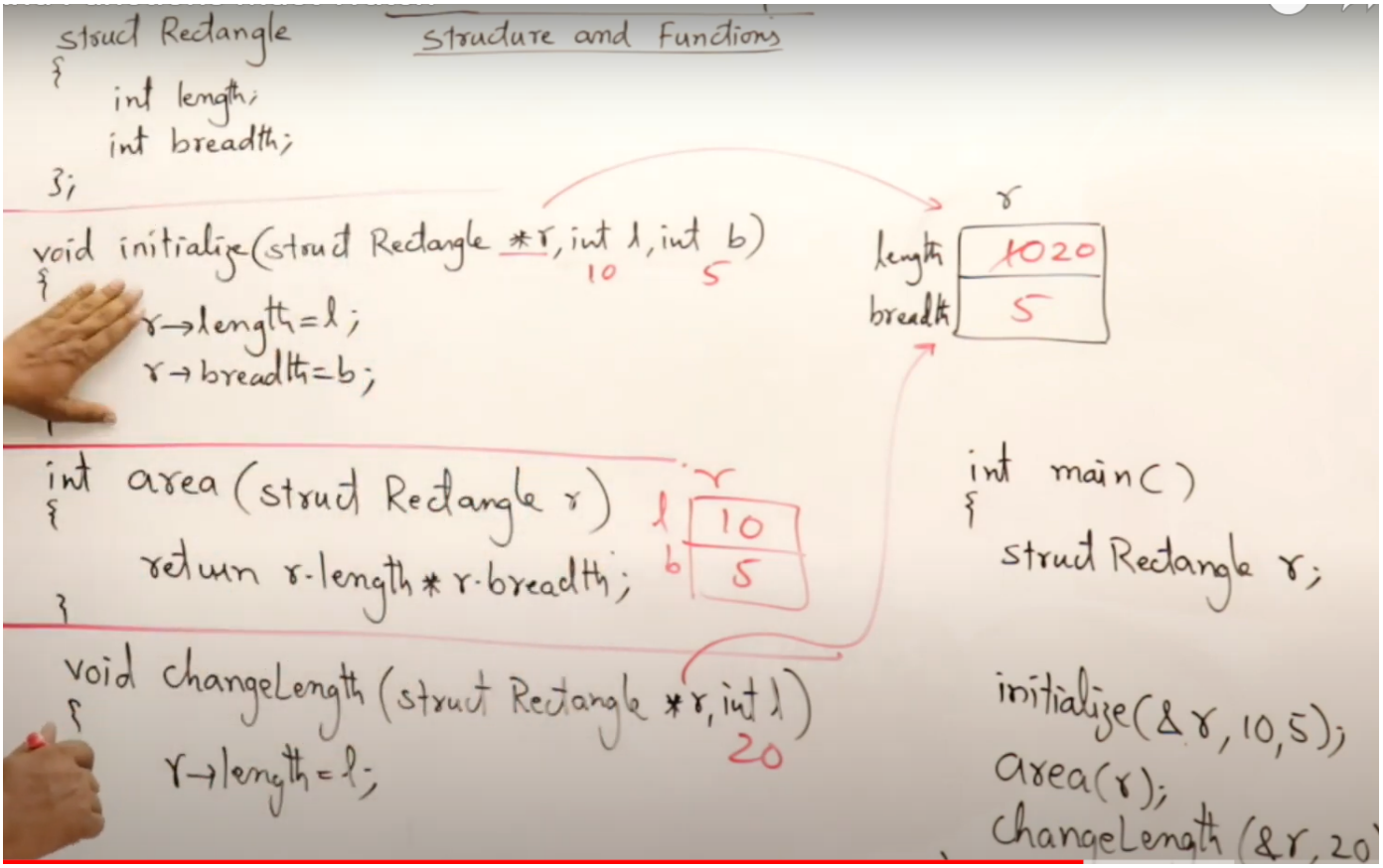
- If you want to pass array by value, pass it by putting it inside a structure.



Returing structure as Parameter -

```
struct Rectangle *fun()
{
    struct Rectangle *p;
    p=new Rectangle;
    //p=(struct Rectangle *)malloc(sizeof(struct Rectangle));

    p->length=15;
    p->breadth=7;

    return p;
}

int main()
{
    struct Rectangle *ptr=fun();
```

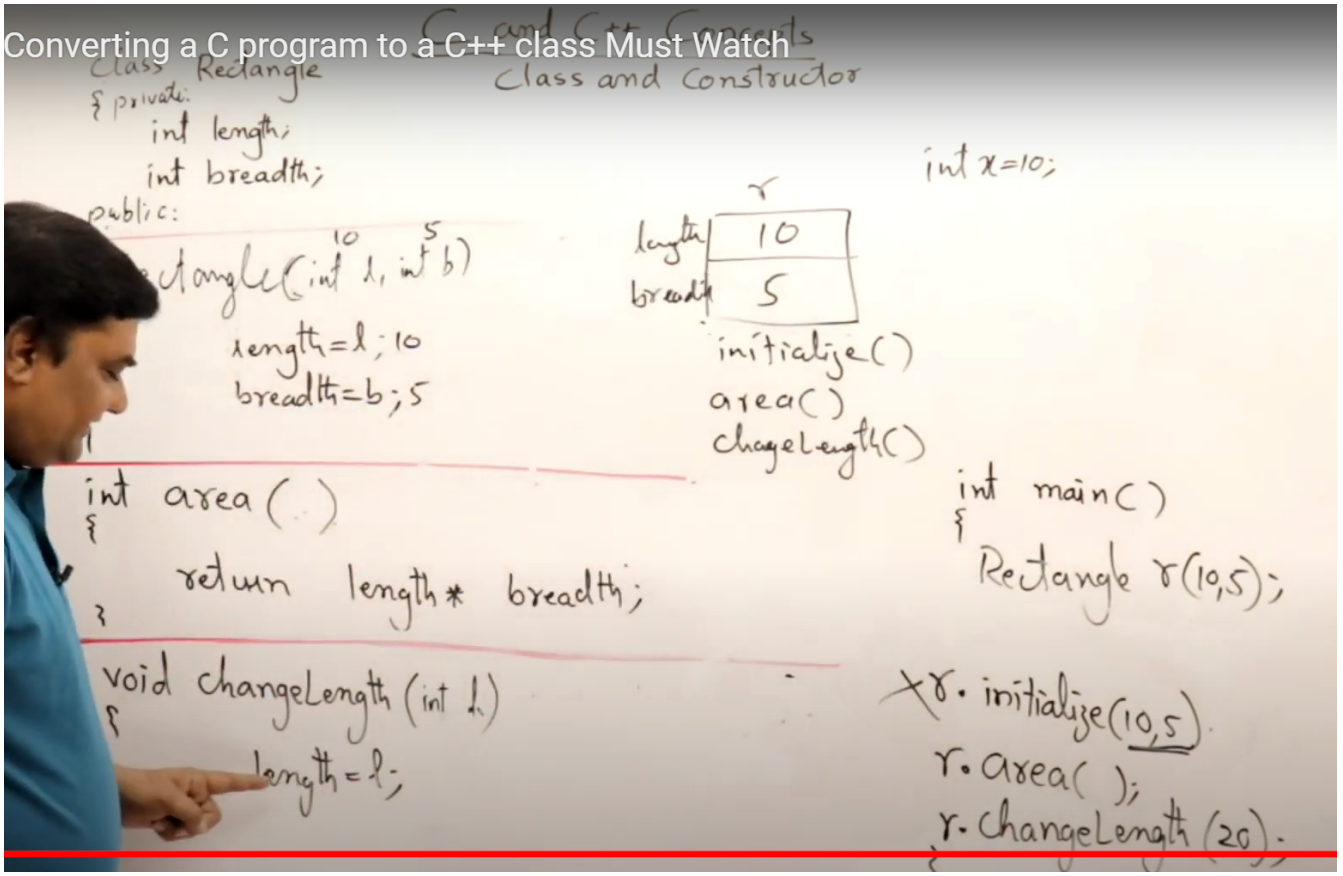
Object Oriented Programming in C -



- How to convert a C Structure into C++ Class -

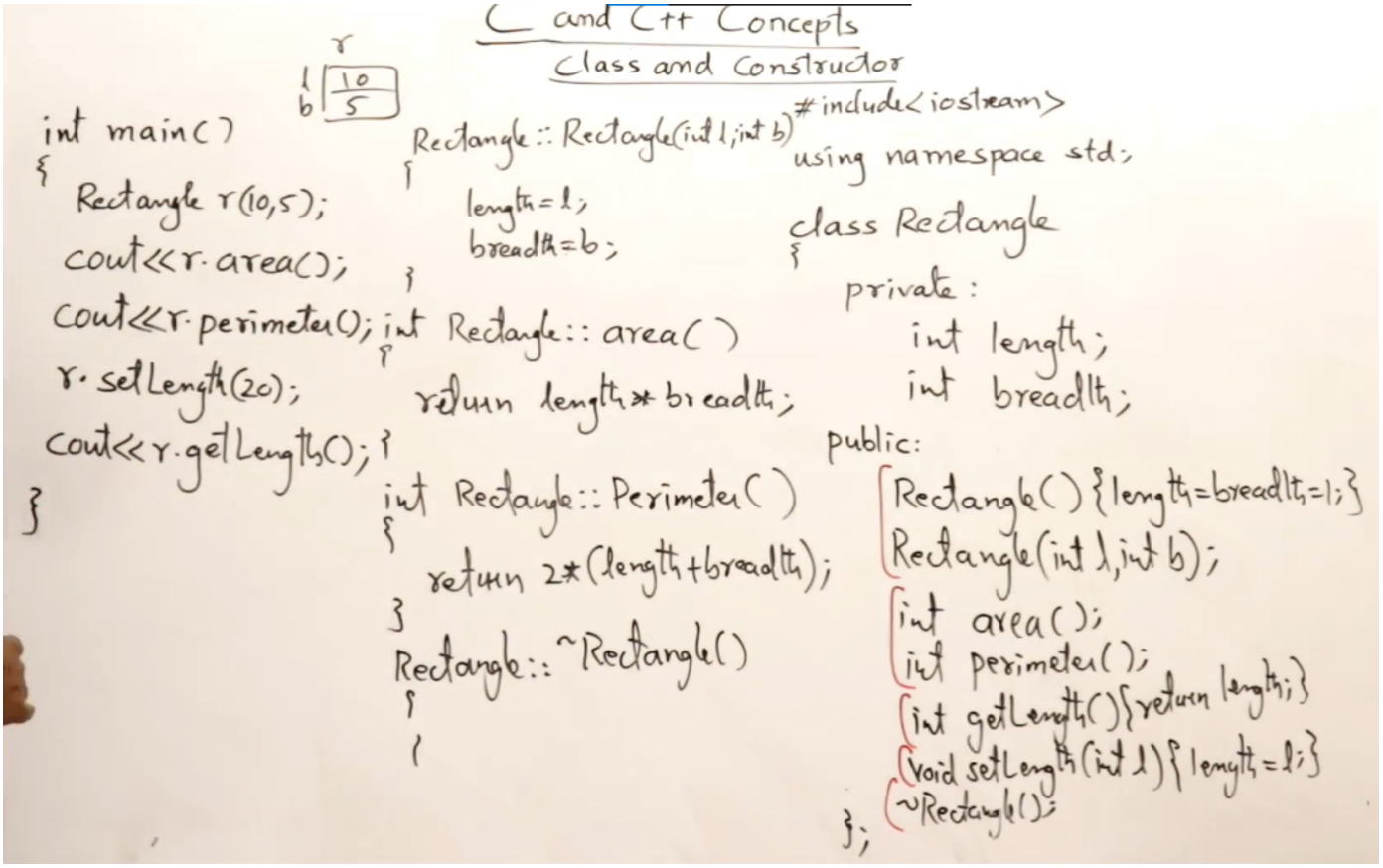
1. struct Rectangle ------ class Rectangle
2. struct Rectangle r --------- Rectangle r
3. Include all functions related to Rectangle inside the class Rectangle.
   Now, since these functions are part of class, no need to give rectangle 'r' as arguement to the functions b/c data members are directly accessible to them.
4. Also, to access data members, no need of arrow or dot operator.
5. Data Members - Private
   Member Functions - Public
6. No need to Pass 'r' while calling these functions, use -
   `r.area or r.changelength`
7. Also, to Initialize an variable of Rectangle classwhile declaring, we can call the constructor -
   `Rectangle r(10,5)`

8. Constructor method is named same as class only, with no return type specified



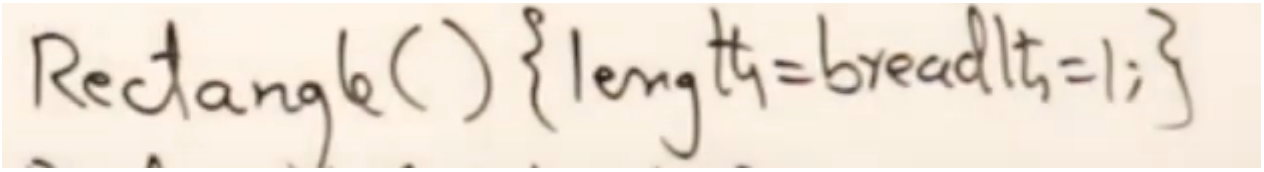
- Always initialize variables
- In structures, everythingh is Public by default and in Class everythingh is by defauly Private.
- An object of an class can only be initialized by a constructor, but you can create an instance tho.

C++ Class -

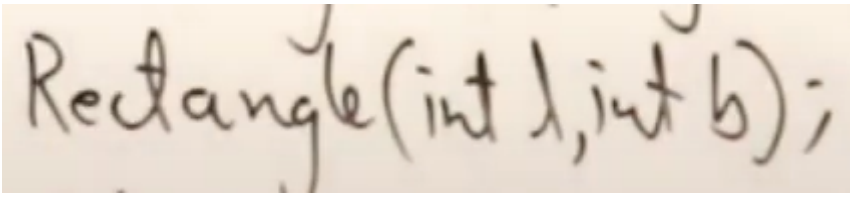


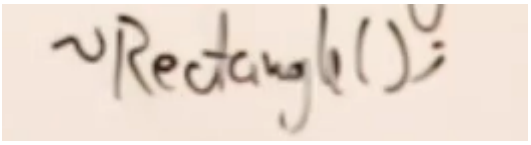
- Constructor - initializes the class
- Overloaded Constructors - means having 2 constructor of same name inside a classs but have different prototype(might be change in numner of i/p parameter)

1. Default Constructor -



2. Parametized Constructor -



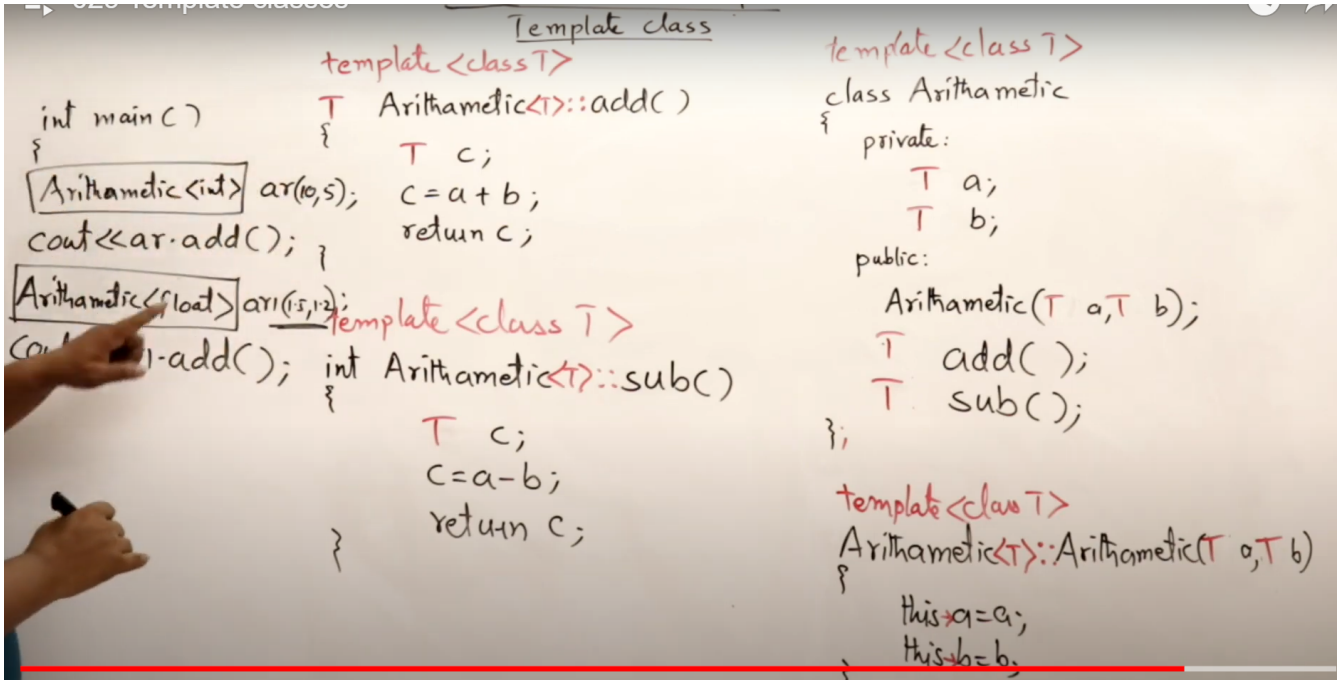


- Facilitators - member functions which perform some operation on data members
- Accessors - the getter function is called accessor. it provides access to data members.
- Mutator - setter function is called mutator. It helps changing value of data members.
- Destructor - Destroy the class/ Removes dynamically allocated memory if any.
  It is written with a tilde symbol in front of class name
  It is Automatically called once the scope of that class object ends(usually when main function ends)

- We can define prototypes of these above function inside the class and then definition of these functions can be defined outside class using scope resolution operator (::)

C++ Template Classes -

- Class defined for one data type may not work for other data type. So, to overcome that we can declare a class as template and replace that particular data type with - `<T>` type.Also, before scope resolution, add
- write `template<class T>` above class declaration and all its associated function.



- Here, "this" is a pointer to current object. So, to access the data members we can use "cap operator(i.e. ->)".
  It is used when data members and constructor parameter have same name.
  Eg - `this->a = a;`
- Inside main function, we can define the data type for current class instance inside <>.
  Eg - Arithmetic ar(10,5)
- We can typecast results in C++ as - (int)

```
int main()
{
    Arithmetic<char> ar('A','B');

    cout<<"Add "<<(int)ar.add()<<endl;
    cout<<"Sub "<<(int)ar.sub()<<endl;

    return 0;
```

- mingw is a compiler.
- To use a debugger, Add a break point and then add variables to wach List.