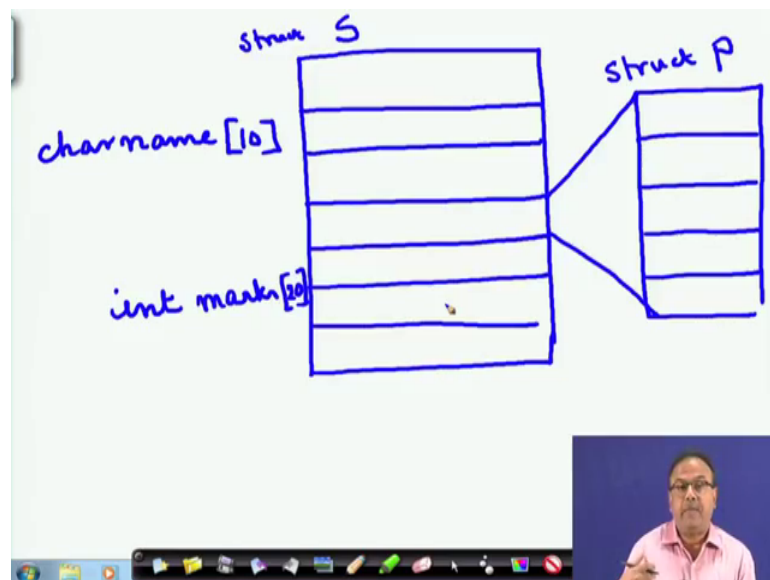


Problem Solving through Programming In C
Prof. Anupam Basu
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 57
Structure with typedef

We were discussing about structures, and in particular we have discussed about the way the structure is formed and in that context we have seen that a structure can contain within itself some members can be themselves a structure right.

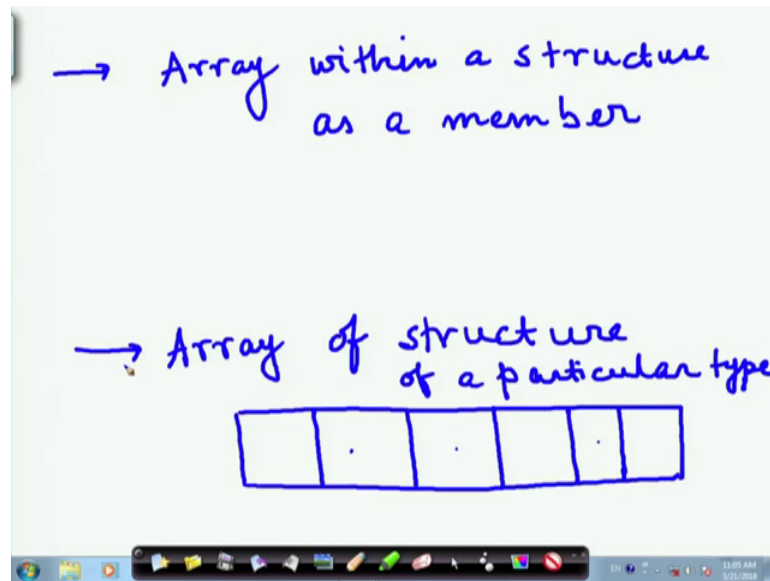
(Refer Slide Time: 00:32)



For example, this one member which can itself be a structure consisting of other members this possible right. So, this is a structure struct S in which there is another struct say P which is a member of this is possible, also some member can be an array this can be say name 10 char; that means, it is a character array of name 10 similarly it could be some member can be int marks, 20 like that it can happen.

Now, so, we have seen that we can have arrays within structure also in the last lecture we have talked about arrays of structure. So, there is a scope of confusion between these two that is why I want to make it clearer that there can be an array within a structure as a member all right.

(Refer Slide Time: 02:08)



So, and the other thing is we can have array of structure. So, in this case we have got an array, every element of that array is a structure of a particular type right. We know that an array can hold data elements of the same type. So, if I have one particular structure defined like say student and it can be an array of students. So, they will be each of them is of the type student and we have also seen.


So, these two must be differentiated very clearly all right.

(Refer Slide Time: 04:03)

Arrays within Structures

- A structure member can be an array:

```
struct student {  
    char name[30];  
    int roll_number;  
    int marks[5];  
    char dob[10];  
} a1, a2, a3;
```
- The array element within the structure can be accessed as:
a1.marks[2]



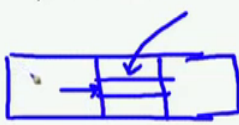
So, as you can see that here a structure, member can be an array we saw it in the last class last lecture that say for example, a structure student has got an array char character array name thirty all right this is a part of an array. Now, that we know arrays within structure, we have also seen arrays of structure, where we can have a number of elements of that particular structure here.

(Refer Slide Time: 04:54)

Arrays of Structures

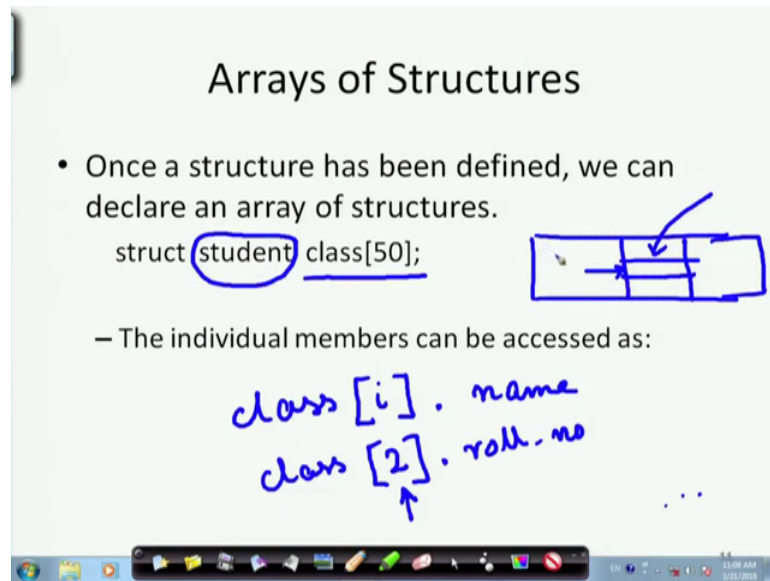
- Once a structure has been defined, we can declare an array of structures.

```
struct student class[50];
```



– The individual members can be accessed as:

class [i]. name
class [2]. roll. no



So, here we have seen that we have got a an array class, the array is class which can hold up to 50 elements and each element of this class is a structure of type student.

So, individual members when I come to say class i then I am actually accessing a structure a particular structure in that array a particular structure in that array. Now that structure has got number of fields therefore, if I have class i dot name, then I will get a particular name if that be the field or I can say class 2 dot roll number.

So, the here I think this distinction must be very clear that there is an array of structures and we can handle them just as here the index is pointing to the particular element, and then by dot operator we are going inside that element and looking at that array.

(Refer Slide Time: 06:32)

Arrays of Structures

- Once a structure has been defined, we can declare an array of structures.

```
struct student class[50];
```
- The individual members can be accessed as:

class[i].name[j] student a1
a1.name[i]

Whereas in the case of an array within a structure we can we look at that in a different way, that is if there be an then I come to the structure; student and suppose there is a variable of variable of name say a 1 as of type student, then I can say a 1 dot say name and I can take a particular character of that.

So, look at the difference of this, this is one the other one was class i dot may be name j. So, here I am accessing the particular element from that structure, and here I am coming to the structure and going to that array element within that structure. So, this is an usual point of confusion among many students that is why I was repeating this.

Next we will start looking at an important aspect an important style of writing programs which is typedef.

(Refer Slide Time: 07:46)

Defining data type: using typedef

- One may define a structure data-type with a single name.
- General syntax:

int
float
char

The slide shows a presentation slide with a title and two bullet points. The word 'typedef' in the title is circled in blue. To the right of the bullet points, the words 'int', 'float', and 'char' are written in blue handwriting. At the bottom right, there is a small video inset of a man in a pink shirt. A Windows taskbar is visible at the bottom of the slide.

That facilitates our programming with structure as the name typedef implies, we are defining some type. We know that int is a type, float is a type, data type char is a type etcetera. Now, I can also define my own type that I will be using in my program using this typedef statement typedef command.

So, we may define a structure a structured data type using a typedef command like this let us see the syntax.

(Refer Slide Time: 08:36)

Defining data type: using typedef

- One may define a structure data-type with a single name.
- General syntax:

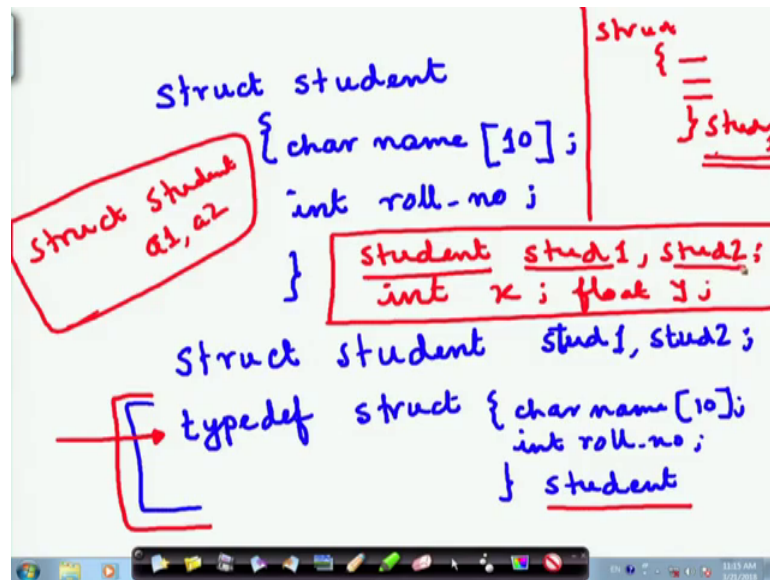
```
typedef struct {  
    member-variable1; char name[.];  
    member-variable2; int roll ;  
    .  
    member-variableN;  
} tag;                      → Students
```

- tag is the name of the new data-type.

The slide shows a presentation slide with a title and two bullet points. The word 'typedef' in the title is circled in blue. Below the bullet points, a C code snippet is shown with handwritten annotations in red. The annotations include 'char name[.];', 'int roll ;', and 'Students' with arrows pointing to the 'tag' in the struct definition. At the bottom right, there is a small video inset of a man in a pink shirt. A Windows taskbar is visible at the bottom of the slide.

The syntax is I say typedef struct member variable 1 member variable 2 member variable n and then some tag is the name of the new data type. Let me illustrate this differentiating this with the earlier thing, say earlier we had written something like this struct student.

(Refer Slide Time: 09:07)



Then char name 10 int roll number like this. So, I was defining a structure in this way, and I was later on referring to the structure as struct student say student 1 student 2 in this way all right. Now, what I am trying to do here is, I am trying to define a type suppose I do not write this what I am doing instead I am saying typedef, struct same thing char name 10, int roll number and name this type as student. If I do this then next time when I write the student declare the student 1 and student 2 I will do it in a different way I can do it much simply as I am writing now, in the red.

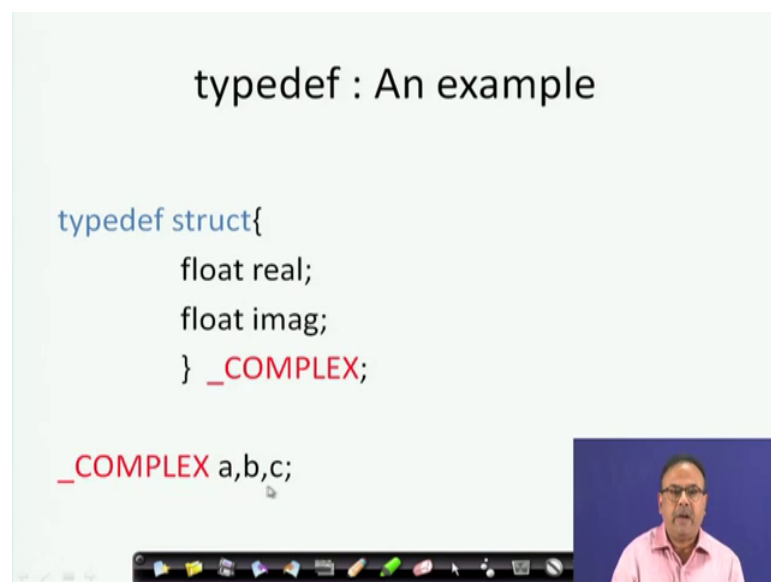
If I just simply right look inside this red box; if I write student student 1 comma student 2 that is sufficient just as compared this with the way we had declared int x float y. So, here int and float were some types, now here student 1 student 2 are two variables of type student. Now, this type is not a default type it is not pre-defined in c therefore, but I can use it because I have already using this typedef statement has defined the structure as student.

But please differentiate between this and the normal definition of the student as we do normally as say struct something and then I say student. Now, this is here is it clear, I say

struct we have seen this and I write the members here and then define that structure as the type student. Now this is a definition once for all a particular structure as student now this, if I have to use it then I have to say struct student followed by a 1 a 2 student 1 student 2 whatever. But in this case if I do this typedef, I am saved from this problem. So, let us have a little look at how we will go about it.

So, that is a tag. So, here typedef a particular structure just as here could be char name or int, roll just like that note that, the semicolon at the end of if all this declarations and ultimately I name this tag semi say for example, here as student. So, tag is the name of the new data type.

(Refer Slide Time: 13:59)



So, given that here is an example again the example of complex number. So, we are defining a type called complex, ignore this underscore here.

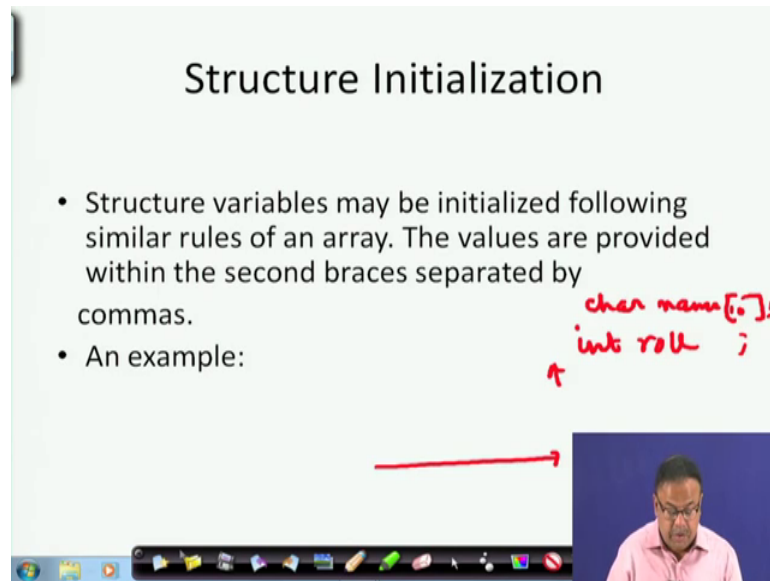
So, typedef something having real part and an imaginary part is known as a data type complex and then I am saying complex a b c just like int a b c float a b c, I can write complex a b c because complex has already been defined here.

(Refer Slide Time: 14:33)

Structure Initialization

- Structure variables may be initialized following similar rules of an array. The values are provided within the second braces separated by commas.
- An example:

```
char name[10];  
int roll;
```



Because complex has already been defined; so, I think this is clear right.

Next we move to, how do we initialize a structure. We saw that we can initialize an array similarly we can initialize a structure as well. Say a structure variables may be initialized following similar rules like an array. The values are provided in a within the second braces separated by commas for example, here complex a.

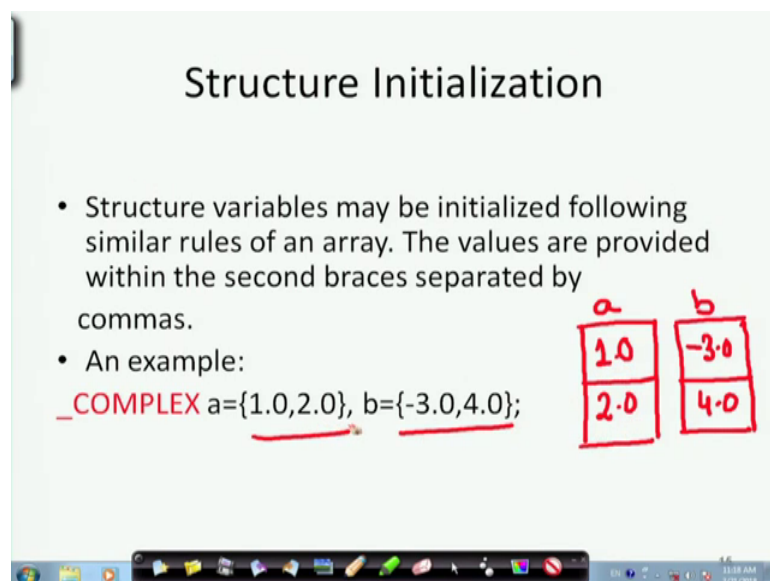
(Refer Slide Time: 15:14)

Structure Initialization

- Structure variables may be initialized following similar rules of an array. The values are provided within the second braces separated by commas.
- An example:

```
_COMPLEX a={1.0,2.0}, b={-3.0,4.0};
```

a	b
1.0	-3.0
2.0	4.0



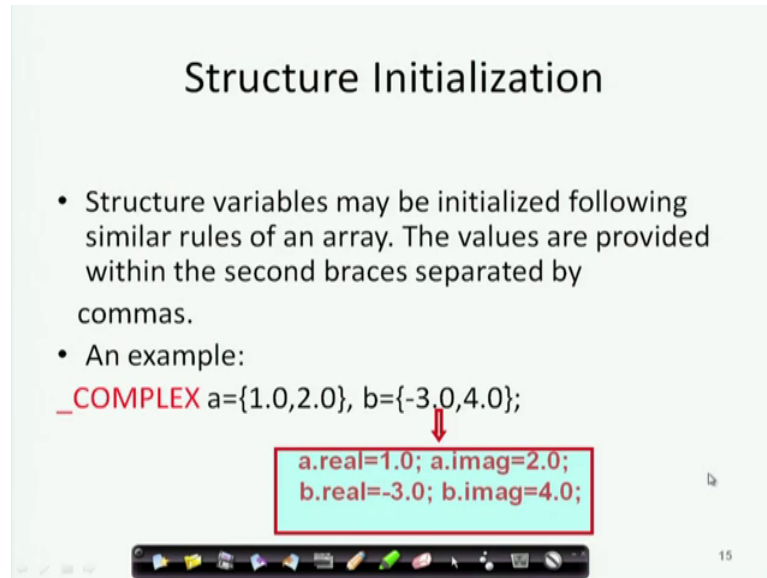
Complex a, is a complex variable 1.0 comma 2.0 what does it imply? It implies that it is a structure having two fields and one field is 1.0 another is 2.0 similarly b is another

field, which is initialized to minus 3.0 and 4.0. Just as we did it in the case of arrays we do it for individual structural variables, but you have to put all the values initial values for all the fields of the array.

(Refer Slide Time: 16:12)

Structure Initialization

- Structure variables may be initialized following similar rules of an array. The values are provided within the second braces separated by commas.
- An example:
`_COMPLEX a={1.0,2.0}, b={-3.0,4.0};`
↓
`a.real=1.0; a.imag=2.0;
b.real=-3.0; b.imag=4.0;`



So, this is what happen by this initialization.

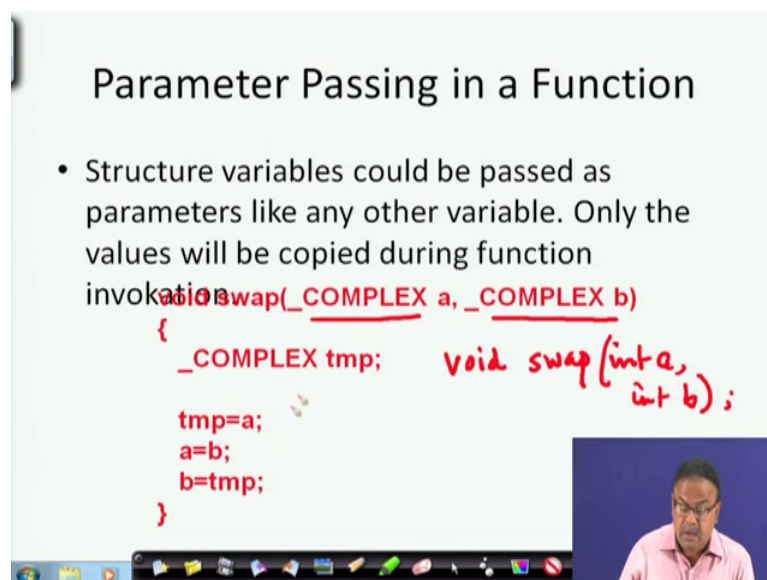
(Refer Slide Time: 16:17)

Parameter Passing in a Function

- Structure variables could be passed as parameters like any other variable. Only the values will be copied during function invocation.

```
void swap(_COMPLEX a, _COMPLEX b)
{
    _COMPLEX tmp;
    tmp=a;
    a=b;
    b=tmp;
}
```

void swap(int a, int b);



The other thing that last thing that we will be talking about is parameter passing in a function. How do we pass a structure as a parameter to a function? We have seen how arrays can be passed similarly how can we pass structures to a function like any other

variable. Just like any other variable we can pass it like for example, here you can see that there is a swap between complex variable a and complex variable b. Now, a and b are both structures. So, I am just saying just as we use to write say void swap(int a, int b) just like that, here I write complex a, complex b ok. It is also a call by value. Now so, here is again the typical assignment and the way we carry it out.

(Refer Slide Time: 17:37)

An example program

```

#include <stdio.h>

typedef struct{
    float real;
    float imag;
} _COMPLEX;

void swap(_COMPLEX a, _COMPLEX b)
{
    _COMPLEX tmp;

    tmp=a;
    a=b;
    b=tmp;
}
    
```

The diagram illustrates the execution of the swap function. It shows three boxes representing complex numbers: 'tmp' (3.0, -2.0), 'a' (3.0, -1.5), and 'b' (5.0, 2.5). Red arrows show the flow of data: 'a' is copied to 'tmp', 'b' is copied to 'a', and 'tmp' is copied to 'b'.

So, here is an example program using typedef. What is being done here is I am defining a type complex, typedef struct they are two parts real and imaginary and this data type is known as complex. Now I am writing a function swap them. So, I am taking another I have taken another variable which is of type complex tmp is of type complex. So, when I do assign a to tmp, then this tmp variable which is of type complex will get will copy the variables say here it was 3.0 and minus 2. So, this will come here it will become 3.0 and this will become minus 2.

So, that is tmp. Now I copy a to b. So, here sorry b to a I am sorry not this one, there was some b and the values of b are copied to a not here this are copied here and here. So, might be this is changed to 5, this is changed to 2 point minus 2.5 and then I copy again b to tmp, tmp to b. So, this is again copied back. So, member by member the copy is done. So, here we illustrate what you mean by typedef and a function that is using it.

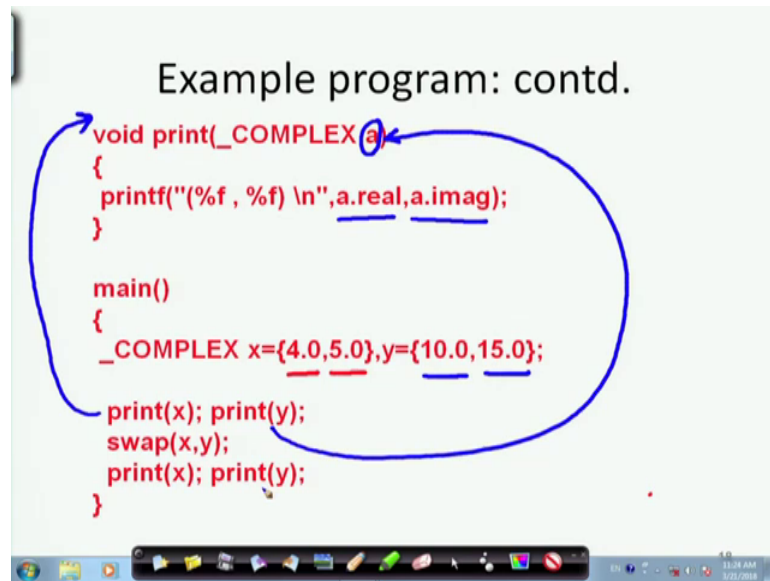
(Refer Slide Time: 19:55)

```
Example program: contd.

void print(_COMPLEX a)
{
    printf("%f , %f \n", a.real, a.imag);
}

main()
{
    _COMPLEX x={4.0,5.0},y={10.0,15.0};

    print(x); print(y);
    swap(x,y);
    print(x); print(y);
}
```



So, now suppose how can we print a structure. Print I am now printing the comp say here is a function the main function is calling print x and print y, the main function is calling print x and print y. So, say for example, my main function is initializing now you understand all this things, x to 4 x real part to 4 imaginary part to 5 right and wise real part to 10 and imaginary part to 15. So, now, I am calling print x. So, it is calling print x. So, x is being copied call by value to another local variable a, which is local to this function print. So, a is holding the complex variable x and print f a dot real a dot imaginary.

So, field by field I print them then I come back I print y. Y is copied to a and the same thing happens then I call swap. Now, swap as a function that we just now saw where we take tmp and we copy it to tmp, and then I copy b to a, and then tmp to a in that way we carry out the swapping.

So, it is purely call by value and then I print x and print y. After I swap what will happen in this case? If I go back to this case where I am passing among this a and b, the x and y are being swapped inside that function, but will that be swapped in actuality you check that yourself.

(Refer Slide Time: 22:10)

x, y

Returning structures

$z = \text{add}(x, y)$

- It is also possible to return structure values from a function. The return data type of the function should be as same as the data type of the structure itself.

```
COMPLEX add (COMPLEX a, COMPLEX b)
{
    COMPLEX tmp;
    tmp.real = a.real+b.real;
    tmp.imag = a.imag+b.imag;
    return tmp;
}
```

Diagram illustrating the addition of two complex numbers:

- Complex number a : real part 3.0, imaginary part 4.2
- Complex number b : real part 2.5, imaginary part 3.6
- Resulting structure tmp : real part 5.5, imaginary part 7.8

The diagram shows the calculation: $3.0 + 2.5 = 5.5$ (real part) and $4.2 + 3.6 = 7.8$ (imaginary part).

How do we return a structure from a function? Suppose I have done something, but in this earlier case x and y were being swapped inside the function, but when I come back to the main my x y remains the same. So, if we if we want to return the function for example, here, I carry out a real plus b real I make it tmp, say here is an example of adding two complex numbers here complex a and complex b are two variables which have been passed on as parameter here, and I have got a tmp as an local function local variable.

So, tmp dot real is copping a dot real plus b dot real. So, what is happening? Here I have got structure a with a real part and an imaginary part. So, 3.0 4.2 another is b which is 2.5 and may be 3.6.

Now, when I am adding two complex numbers you remember we discussed it in last class also, I first add this two. So, a real is added to b real. So, I am getting in tmp, this is tmp where I am getting 5.5 and then imaginary I am getting 7.8 and now I return tmp.

Now in the main function if I had written something like say x and y were two complex variables, and what I did I said add x y assign to some other complex variable z , in that case the tmp this tmp will be assigned to this complex variable z that is how we can return a structure from a function.

(Refer Slide Time: 24:52)

Returning structures

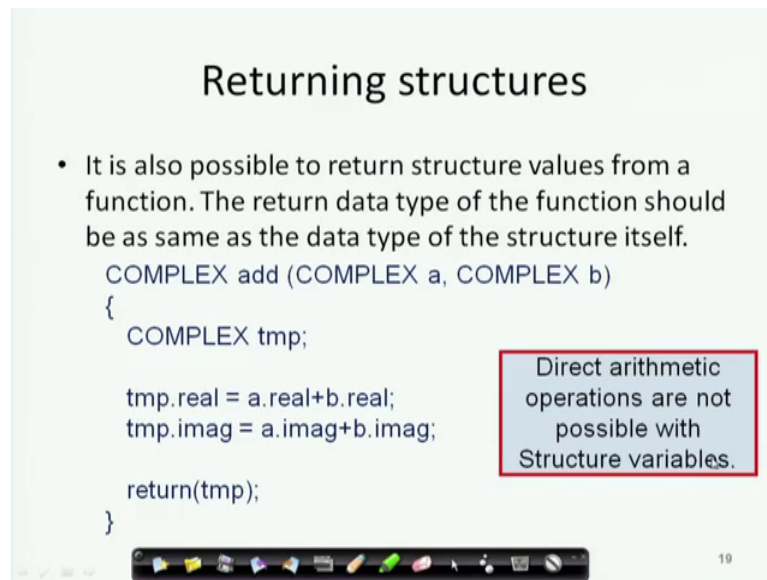
- It is also possible to return structure values from a function. The return data type of the function should be as same as the data type of the structure itself.

```
COMPLEX add (COMPLEX a, COMPLEX b)
{
    COMPLEX tmp;

    tmp.real = a.real+b.real;
    tmp.imag = a.imag+b.imag;

    return(tmp);
}
```

Direct arithmetic operations are not possible with Structure variables.



So, structure facilitates in many ways in this. So, direct arithmetic operations are not possible with structure variables; that means, I cannot just add a and b when both of them are structures, I have to do the arithmetic operations over its members. So, with that we conclude our discussion on structures and a structure is a you will be given assignments on structures, and this will enable you to write different types of handle different types of data types together and using the typedef, you can design your own complex data type which you can utilise further.

So, please practice using structures it is not at all difficult, only a little practice and a little understanding is required.

Thank you.