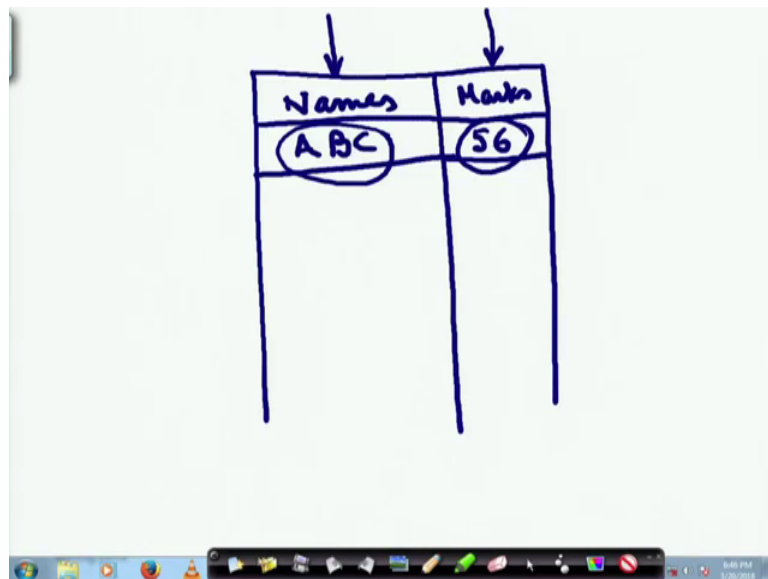


Problem Solving through Programming In C
Prof. Anupam Basu
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 55
Structure

Earlier, we had talked about storage of data in arrays. Now, what we found; what we mentioned in the case of arrays is that in an array, we can store data only of a particular type; particular say an array of integers or an array of floats or an array of characters, but we could not mix different types in the same array. For example, we had faced the problem of representing the student's database where we will have the student's name as well as the student's marks.

(Refer Slide Time: 00:57)

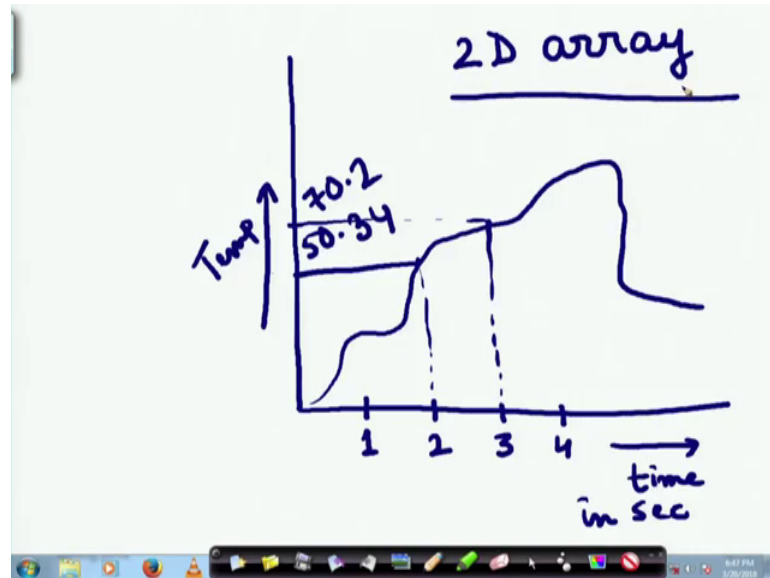


So, we needed something like this; actually that is what was desired; that will have the names here and names will be nothing, but an array of characters. So, here will be some names ABC and here will be the marks 56. Now this representation requires two arrays; one is the representing the names as one array and representing marks as another array.

Now; however, as the diagram is showing here, we are trying to represent them in an unified structure in the same structure, but that is not possible in the case of an array, here we have got 2 different data types. This is an array of characters and this is an

integer, all right. For example, similarly we can see that if I had stored something that say.

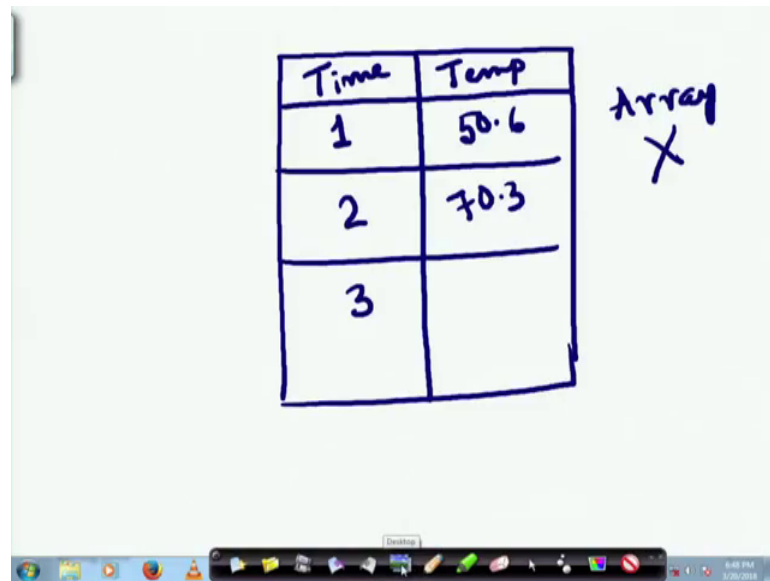
(Refer Slide Time: 02:05)



For example, I want to store a graph, all right, all right, a graph where at every point; 1, 2, 3 seconds; for example, this time in second and here I am measuring the temperature and if I have a graph which is something like this, I cannot say that at every point, the temperature will also be an integer therefore, if I had tried.

So, suppose here at point number 2, the temperature is 50.34 at point number 3, the temperature can be 70.2. So, if I had tried to represent that in the form of a 2D array, in that case, it was not possible because a 2D array is also an array and therefore, is of only one type we have to declare a 2D array as an array of integer or as an array of characters or array of float. So, I cannot represent that in a 2D array where one side.

(Refer Slide Time: 03:38)



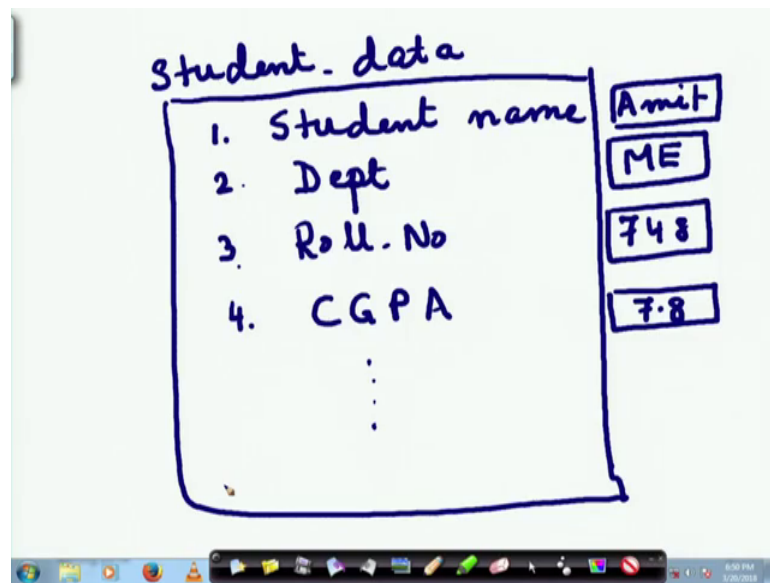
Time	Temp
1	50.6
2	70.3
3	

Array
X

Say for example, one will be that one column will be temperature and another column will be the time and temperature. So, that was not possible to be done in a 2 dimensional array because time while time will be an integer 1, 2, 3 temperatures can be something different 50.6, 70.3, etcetera. So, an array is not possible.

So, our question is then what is the type of data structure; what is the type of arrangement by which I can represent data of different types together and the answer to that is what we will be looking at today that is structure structures in C allows us to represent a combination of different data types.

(Refer Slide Time: 04:53)

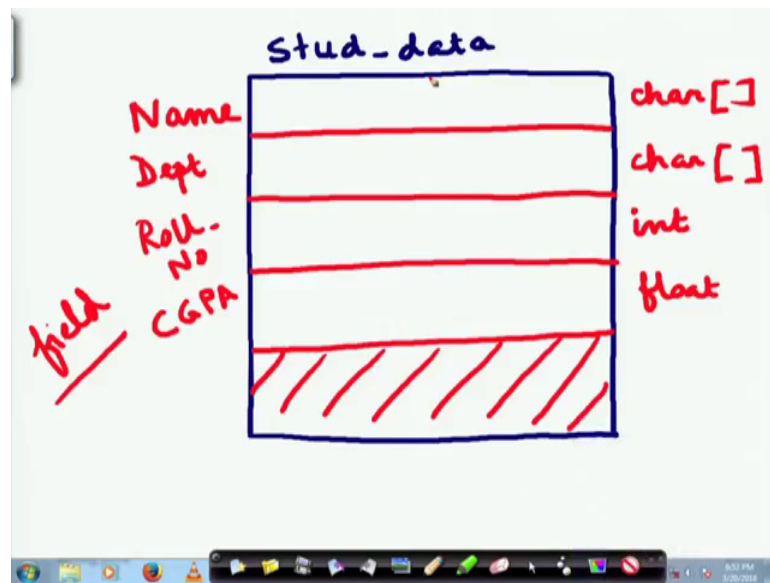


For example, let me give an example of a structure, say I define I want to define student data and student data will consist of the student name, say the department of the student, roll number of the student, may be the CGPA of the student, grade, point, average of the student, etcetera.

Now, let us look at; if I want to store them in store them together as a common piece a student data, then this entire thing, the student data has got components of different data types; for example, what will be student name student name, say for example, Amith will be either a string or an array of character department, somebody can say mechanical engineering that can also be a character string, whereas, roll number 748 will be an integer and CGPA say 7.8 will be a floating point number.

So, what we can see that in order to store the student data we have to have a mixture of different data types. So, a structure allows us to do exactly this where I can consider this student data as a structure. So, let me redraw it in a different way.

(Refer Slide Time: 06:44)



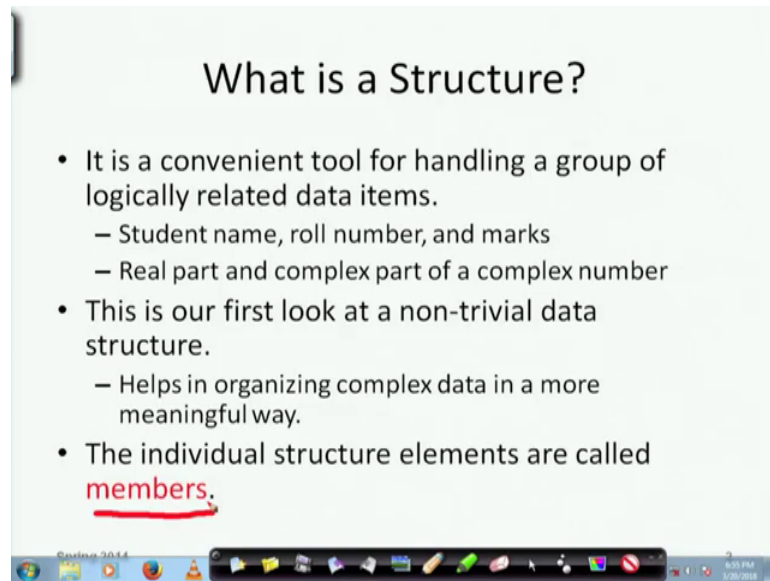
So, I draw a box now which is representing the student data, I name it stud data and I have got different fields in this one field and each field has gotten some identification. So, this field is storing the name and this field is basically a character adding a type of type character array. So, another field is name is department and its type is also character array another field remember that I every time I am using that term field of student data.

So, name department what else it you have role number which will be of type integer and CGPA which will be float suppose there is no other field alright. So, we call this entire box to be a structure this is a structure and what is contained in this structure there are fold 4 fields that define the structure the 4 fields are name department roll number and CGPA, these 4 fields are defining a structure. Now C allows us to define structures of this type and we learn how we can define such things and let us go ahead a little bit, what is the structure? It is a convenient tool for handling a group of logically related data items here.

(Refer Slide Time: 08:55)

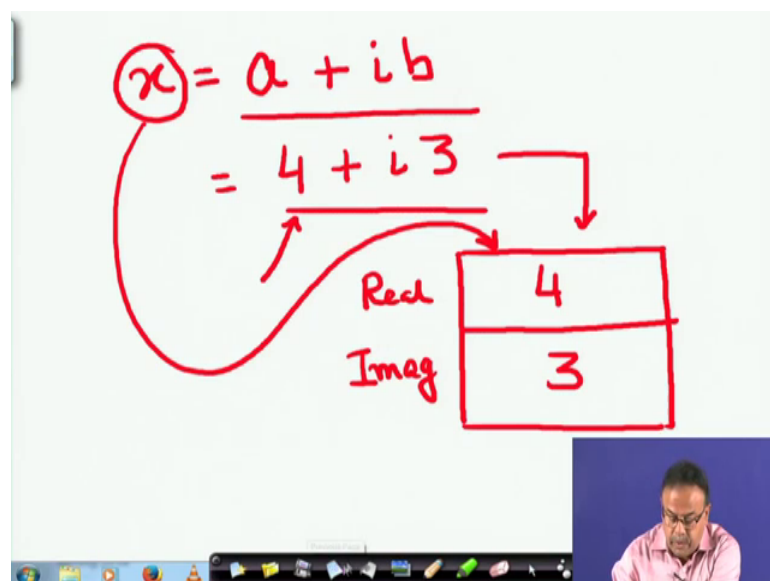
What is a Structure?

- It is a convenient tool for handling a group of logically related data items.
 - Student name, roll number, and marks
 - Real part and complex part of a complex number
- This is our first look at a non-trivial data structure.
 - Helps in organizing complex data in a more meaningful way.
- The individual structure elements are called members.




Of course, the logically related data items were the different information fields or information component that are related to a particular student. So, till now, we saw very simple data structures like array. Now here for example, I have got related components student name roll number and marks or CGPA or for example, a real part and complex imaginary part of a complex number, for example, if I had tried this, say for example, all of you know that a complex number he is told as a plus ib , right that is a complex number having.

(Refer Slide Time: 09:43)



$x = a + ib$
 $= 4 + i3$

Real	4
Imag	3



So, this is x where x is a complex number $a + ib$. So, maybe x is equal to $4 + i3$ all of you are aware of that; now how do I represent I know how could I can represent an integer, I know, how I can represent a floating point number, but now the question is; how I can represent such a complex number now; obviously, I can immediately think of that; well let me have a structure like this which will have 2 fields one is the real part, there is a real part. So, I call it the real part another part is the imaginary part and I can store this number simply as $4 + 3i$.

So, the imaginary part is 3 and the real part is 4. So, the complex number instead of being an integer or a float is essentially a structure all right this is another very common and easy use of structures that you can see. So, whenever I refer to this variable x which is of type complex, I will be referring to this structure and not neither at 4 or nor at 3 has been looking at this whole thing.

So, so, it helps us in organising the complex data in a much more meaningful way the individual structure elements that I was talking of which is which has may mentioning as fields is also known as members now you will find other names to structures also. So, structures are sometimes called records and each of these horizontal boxes that we are drawing were fields of that record or in terms of strain we referred here to it a structure we call each of those smaller boxes as members of that structure.

(Refer Slide Time: 12:12)

Defining a Structure

- The composition of a structure may be defined as:

```
struct tag {  
    member 1;  
    member 2;  
    :  
    member m;  
};
```

Handwritten notes:

- Struct stud_data* (with an arrow pointing to the structure definition)
- { name*
dept
roll
} cgpa

Explanatory text:

- **struct** is the required keyword.
- **tag** is the name of the structure.

Spring 2014 3

So the composition of a structure can be defined as follows see a structure I can put a tag now this is very important; what is the tag that I am putting over here.

Say for example, earlier I was talking of student data. So, my tag could be stud data as I had written. So, I have to write struct stud data and inside that I have got several members and what well my members my member was name my member was roll number department roll number and CGPA. So, I had 4 members of this now each of these were of different types that is something which is very important to understand. So, struct is the required keyword I must write this word struct. So, that I can really show that it is what this tag is just the name is the name of the structure like stud data. So, that is a structure just as we had done for other variables we had done say int x float y, etcetera.

(Refer Slide Time: 13:46)

Defining a Structure

- The composition of a structure may be defined as:

```
struct tag {  
    member 1;  
    member 2;  
    :  
    member m;  
};
```

— **struct** is the required keyword.
— **tag** is the name of the structure. **book** { =
— **member 1, member 2, ...** are individual member declarations. }

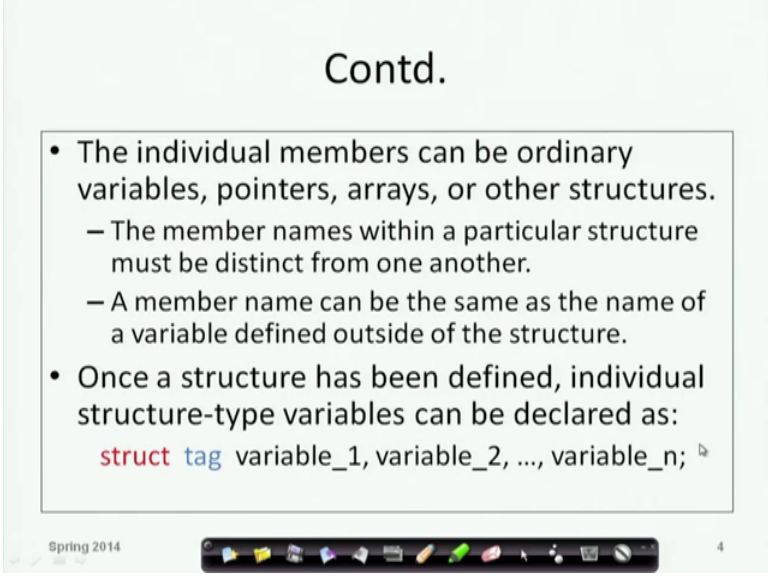
Handwritten notes in red: `int x;`, `float y;`, `struct book`, and a diagram of a structure with three horizontal bars representing members.

Spring 2014 3

So, here also instead of int or float I have to say struct and what the struct is that is defined. So, first of all I have to define the structure int and float are defined by default any c compiler will understand what int or float is, but if I write struct, then by default the compiler will not understand what the structure is because there can be different structure one structure for student names one structure for book details if I had thought of a book detail, I want to store that in a structure what will the components be what will the members be can we just think of the members will be of course, the title of the book the author of the book the publisher year of publication the ISBN number, etcetera.

So, therefore, each of these members will again be of different types. So, it will be a heterogeneous organization just as it was with the student data. So, we have to write if I want to define books for example, I would have to write struct book followed by the definition of the members. So, this is equivalent to as if I am drawing a structure and its name is becoming book and I am defining, what are the members of this structure, alright, let us move a little ahead.

(Refer Slide Time: 15:48)



Contd.

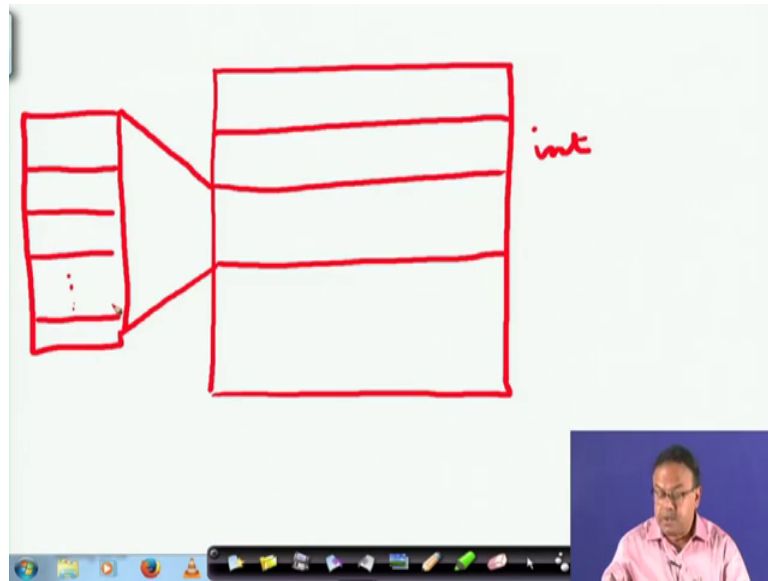
- The individual members can be ordinary variables, pointers, arrays, or other structures.
 - The member names within a particular structure must be distinct from one another.
 - A member name can be the same as the name of a variable defined outside of the structure.
- Once a structure has been defined, individual structure-type variables can be declared as:

```
struct tag variable_1, variable_2, ..., variable_n;
```

Spring 2014 4

So, member 1, member 2, etcetera are individual member declarations will see more examples of this, the individual members can be of different types and that is the beauty of structure the individual members can be ordinarily variables pointers arrays or other structures it can even be other structure that is very important it can be it can be some variable alright just as name or it can be an array say name was a character array or it can even be a structure. That means, I can have a structure within a structure. So, it is possible that I will have something like this that I have got a structure.

(Refer Slide Time: 16:29)



One field of the structure is a character array another field of the structure can be an integer variable or another field of the structure can be a structure itself it can be another structure where there are more members inside that is also possible ok.

So, that is the beauty and the flexibility that we get from structures. So, the member names within a particular structure must be distinct from one another we cannot put the same name to 2 different structures once a structure has been defined individual structure type variables can be declared as we will see struct tag variable one variable 2 variable n, I think, it will be much more clearer, if we go through an example, let us look at this example A.

(Refer Slide Time: 17:44)

Example

- A structure definition:

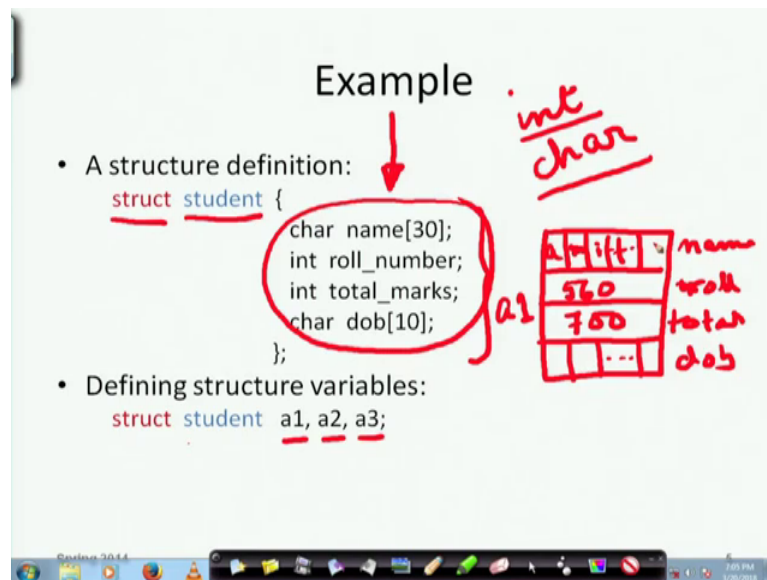
```
struct student {  
    char name[30];  
    int roll_number;  
    int total_marks;  
    char dob[10];  
};
```
- Defining structure variables:

```
struct student a1, a2, a3;
```

Handwritten notes:

- int* (above `int` in the definition)
- char* (above `char` in the definition)
- a1* (next to the first variable definition)
- A table representing the structure data:

<i>Amit</i>	<i>name</i>
<i>560</i>	<i>roll</i>
<i>700</i>	<i>total</i>
<i>...</i>	<i>dob</i>



Structure definition first I start with a structure definition `struct student` and then I put just 2 parentheses to show that there will be some members defined inside.

Now, how do I define the members I say that name one member is name which is an array of character all right the next one is roll number which is an integer. So, roll number is a variable as I said that a structure can field can be variable or can be array or can be another structure. So, here is an example where you have an array and a variable of type `int` again, total marks can be integer data of birth can be character and a sizes. So, data birth is again of type array character array because the date of birth can be say 10 Jan, etcetera, 2010. So, like that it can be a character array.

So, how do you. So, this is how a structure is defined all right. Now, I am now defining the variables, I have defined the structures here. So, I know once the compiler reads this and finds that well I have got some character some definition of some variable called `student` I know that that is a structure and what is the constituent of that structure this is these fields.

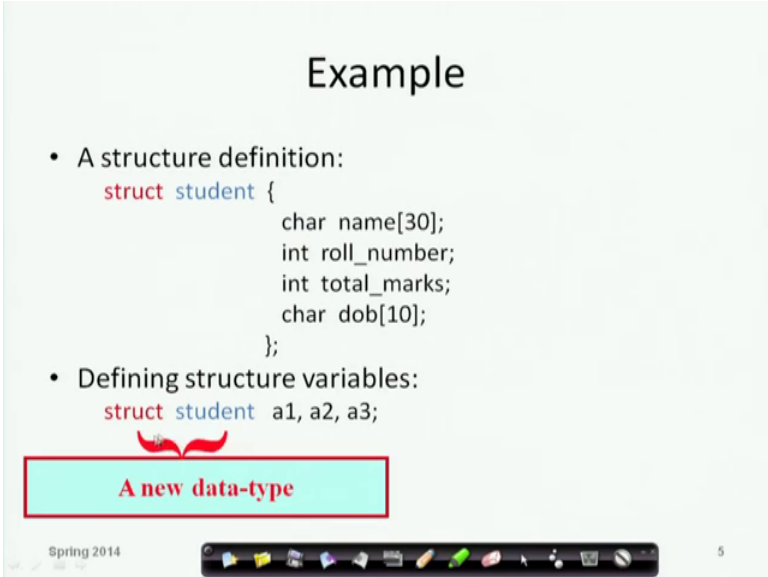
Now this is just a type just as we had `int` `float`, etcetera or `char` those who are data types. Now, I am saying that I have got a structure of type this where there are 4 fields, but the data has not yet been put this is very important though, there are 4 fields one field is name and I say that is an array of thirty characters and there is a role number roll which

is an int total marks is an integer and date of birth is an array, but you see that there is no data already put inside this array.

Now, we have 2 named variables a 1 a 2 a 3 are variables of type student structure and what is that student structure this is a structure. So, a 1 will just be a copy of this structure and a 1 a variable, a 1 may have if I take this may have some name like a math here alright roll number may be 560 and total maybe 700 and date of birth can be something that is one particular instance of this particular structure this particular structure that has been defined this part must be very clear to all of you.

So, I have defined 3 variables a 1, a 2 and a 3 each of type; what type structure student.

(Refer Slide Time: 21:35)



Example

- A structure definition:

```
struct student {  
    char name[30];  
    int roll_number;  
    int total_marks;  
    char dob[10];  
};
```
- Defining structure variables:

```
struct student a1, a2, a3;
```

A new data-type

Spring 2014 5

So, type is struct student, we have got these fields as its constituents ok; so struct student tells us a new data type and these are the variables of the data type.

(Refer Slide Time: 21:38)

A Compact Form

- It is possible to combine the declaration of the structure with that of the structure variables:

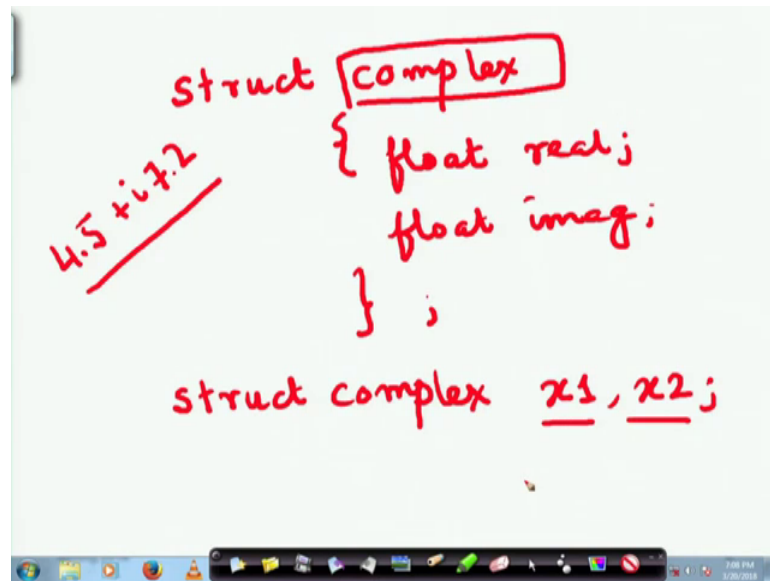
```
struct tag {  
    member 1;  
    member 2;  
    :  
    member m;  
} variable_1, variable_2, ..., variable_n;
```
- In this form, "tag" is optional.

Spring 2014

So, it is also possible to write it in a much more compact form like. So, struct tag and then so, I will write struct student and I declare care name for t int roll no role int total marks care data birth 10 and then I could have put the variables a 1, a 2, a 3, inside this note that here the semicolon is coming here that is that is the end of the definition.

So, here I have done it in a much more compact form compared to the earlier one in this form of course, this tag is optional. So, what I mean is see I define once again give an example to you say a complex number I can write is adds struct complex and members are int need not be always integers.

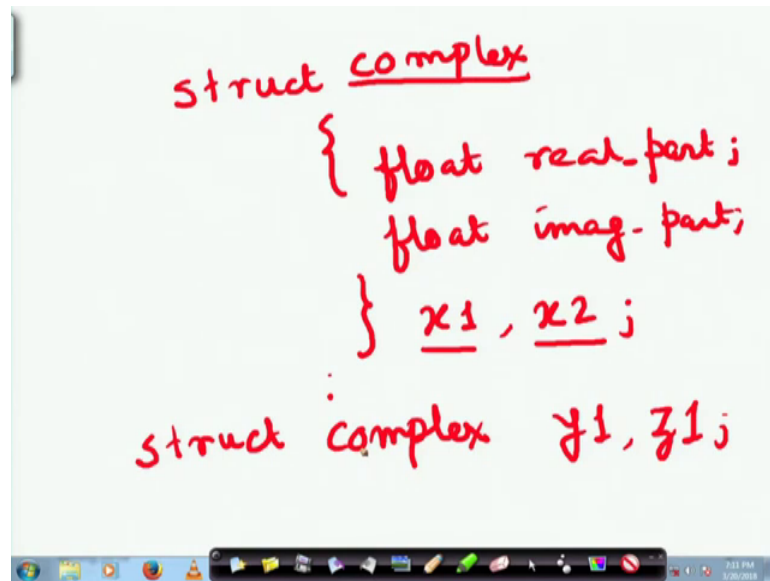
(Refer Slide Time: 22:40)



So, because if I want to store data of type say data of type 4.5 plus i 7.2, then I want to make float real float imaginary all right and that is the end of the structure and I can say struct complex x 1 comma x 2, this is one way of representing structure.

What does it mean that I have first defined the structure called complex which will have these 2 components and then I define that x 1 and x 2 are 2 variables of type this complex or I could have written it also now if I do it in this way I have got an advantage that later on, I can also once I define it I can define some other variables also y 1 y 2 over here or I can also define it as struct float real part.

(Refer Slide Time: 24:36)



```
struct complex
{
    float real-part;
    float imag-part;
} x1, x2;

struct complex y1, z1;
```

Let me in order to do that you do not face confusion, let us write real part float imaginary part and then I can say x 1 comma x 2 note that here, I have not written the thing complex, I have not written that I could have written, it would not have affected me, in any adverse way, but it is not necessary to write it over here. So, if I write this how will the compiler interpret it always you should think what will the come when the compiler looks at it what will it think what would it possibly do.

So, the compiler is trying to understand ok. Now it is there is no ambiguity because it knows that x 1 and x 2 are 2 variables of type structure and the structure is defined here itself I need not give any name, but the advantage of giving a name is say I give the name complex here I have defined x 1 and x 2 later on also say for example, if I had done in this way I could have later on down the line in this program I could have written struct complex say y 1 z 1 its possible, because I have already defined struct complex ok.

So, there is a choice over here which you can think of let us go ahead a little bit. So, in this case the tag is optional.

(Refer Slide Time: 26:40)

Example

```
struct student {
    char name[30];
    int roll_number;
    int total_marks;
    char dob[10];
} a1, a2, a3;
```

Equivalent declarations

```
struct {
    char name[30];
    int roll_number;
    int total_marks;
    char dob[10];
} a1, a2, a3;
```

Spring 2014 7

So, here is an example we will conclude this lecture with this example. So, struct student character name thirty int roll number in total marks character data birth 10 a 1, a 2, a 3; that means, a 1, a 2 and a 3 are 3 students student information 3 pieces of student information or I could have done it in the way I have just now shown the same thing without giving any tag over here these are equivalent.

(Refer Slide Time: 27:09)

Processing a Structure

- The members of a structure are processed individually, as separate entities.
- A structure member can be accessed by writing

• Examples:

Spring 2014 8

Now, next lecture, we will talk about how do you process a structure that will come in the next lecture.