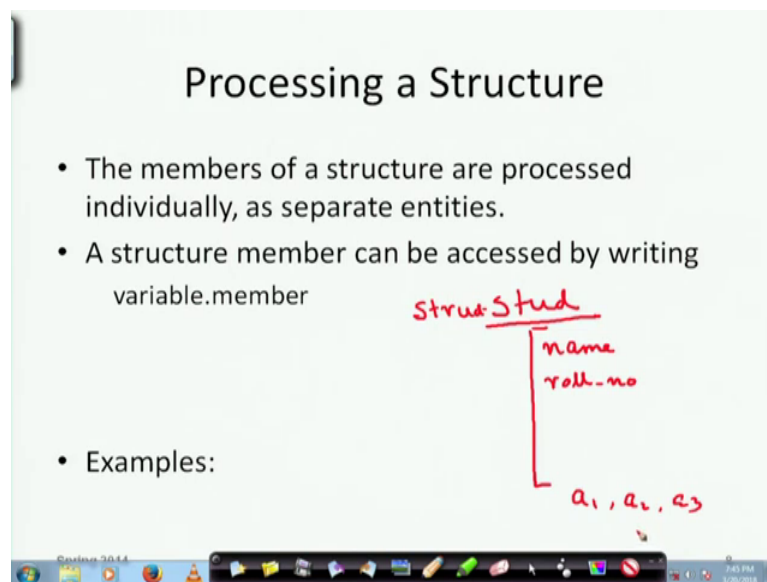


Problem Solving through Programming In C
Prof. Anupam Basu
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 56
Structure (Contd.)

So, in the last lecture we had seen how a structure is defined. Now we will see more examples of that in the course. So, today we will be concentrating on how a particular structure can be processed. In order to process a structure, we can we have to process every field of the structure we can operate on every field of the structure.

(Refer Slide Time: 00:42)



Processing a Structure

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

Handwritten diagram:
`struct Stud`
├── name
└── roll-no

`a1, a2, a3`

So, we can see that a structure how can we access the member of a structure. Say for example, there is a variable like name. So, we can say overalls name of the structure is say student all right; and in the inside student we have got name, roll number etcetera now I can. So, my variable is say all right let me say I have defined like this `struct student`, then all these then the variables are `a1, a2, a3` I am sure you are getting confused a little bit.

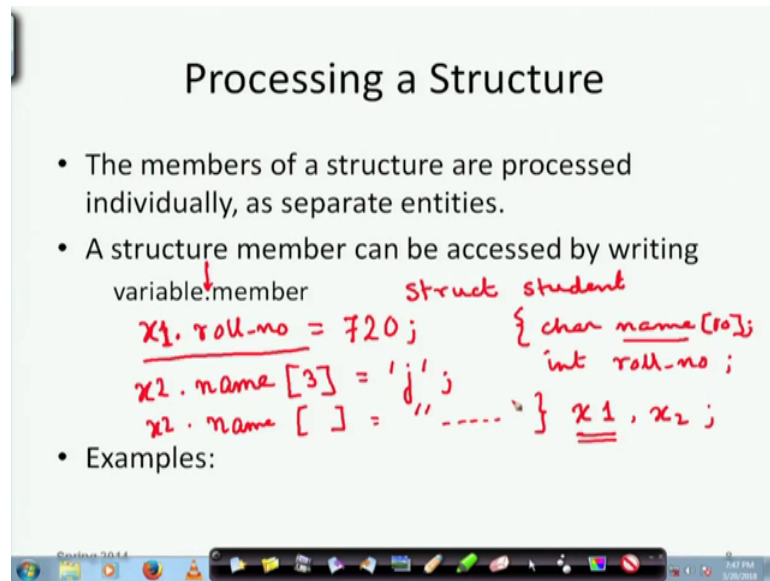
(Refer Slide Time: 01:40)

Processing a Structure

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

```
variable.member      struct student
x1.roll-no = 720;    { char name[10];
x2.name[3] = 'j';    int roll-no;
x2.name[ ] = "-----" } x1, x2;
```

- Examples:



So, let us define it in the proper way as we are done in the last class. Suppose I define struct student, char name 10, int roll number all right suppose only two fields are there and I also say that the variable is variables are x 1 comma x 2 these are 2 variables of the type of this structure whose name is student. So, what is my variable named? x 1. So, here I can say x 1 x 1 dot this is the dot operator which tells about the member, now which member say I want to get the roll number. x 1 dot roll number assign 720 all right. So, this is how I can get access to a particular member of a particular structure. So, I can also similarly write x 2 dot name.

Because that members name is name right name 3 is j all right because name has got a 10 field character ok. I could have written x 2 dot name and I could have assigned a string to this all these are possible.

(Refer Slide Time: 03:46)

Processing a Structure

- The members of a structure are processed individually, as separate entities.
- A structure member can be accessed by writing variable.member where **variable** refers to the name of a structure-type variable, and **member** refers to the name of a member within the structure.
- Examples:
 - a1.name, a2.name, a1.roll_number, a3.dob;

Spring 2014 8

So, if this is clear then we go ahead a little bit were variable refers this variable is referring to is referring to the name of a structure type variable and member is the referring to the field or the particular member element ok. The examples are a 1 dot name, a 1 dot roll number, a 3 dot date of birth etcetera.

(Refer Slide Time: 04:12)

Example: Complex number addition

```
#include <stdio.h>
main()
{
  struct complex
  {
    float real; ←
    float complex; ←
  } a, b, c;

  scanf ("%f%f", &a.real, &a.complex);
  scanf ("%f%f", &b.real, &b.complex);

  c.real = a.real + b.real;
  c.complex = a.complex + b.complex;
}
```

```
printf ("\n %f+ %fj", c.real,
        c.complex);
```

$$x = a + ib$$
$$y = m + im$$
$$x + y = (a + m) + i(b + m)$$

Spring 2014 7:51 PM 1/20/2018

So, now here let us take the old example that we had seen that is a complex number and how we can perform complex number addition. Now, just to refresh your mind say I have I have got two complex numbers x and y and I want to perform x plus y. Suppose x

is a plus i b and y is m plus i n then x plus y is the real parts are added separately. So, it is a plus m plus i then the imaginary parts are added separately, b plus n that we know from our school level knowledge.

Now how can I perform this addition using the concept of a structure. So, here you see in the program, we define a structure called complex and there are three variables a, b and c of this structure type of type complex all right. Now, what are? There that components are one is float sorry I am sorry one is real part, another is I should rename this is actually the imaginary part its renamed in this way here in this program it would have been better, if we I had written float real part float imaginary part however.

Now, scan f I am reading the two numbers a is a variable, a is real part is being read and a is complex part is being read note the format statements because both of them are float we are putting percentage f all right. And so, I read a. So, you see normally I would have just done scan f amperes and a, but here I have to do separately I have to read the real part and the imaginary part, then the sum is being stored in c.

So, c is real will be c is the sum. So, it is real will be sorry it is a real part will be the real part of a plus the real part of b and the imaginary part or the as it is written here complex part is the complex part of imaginary part of a plus the imaginary part of b. Then I am printing look at how I am printing it, printing percentage f the real part of c and plus then I put j and then the complex part of c.

(Refer Slide Time: 07:19)

Example: Complex number addition

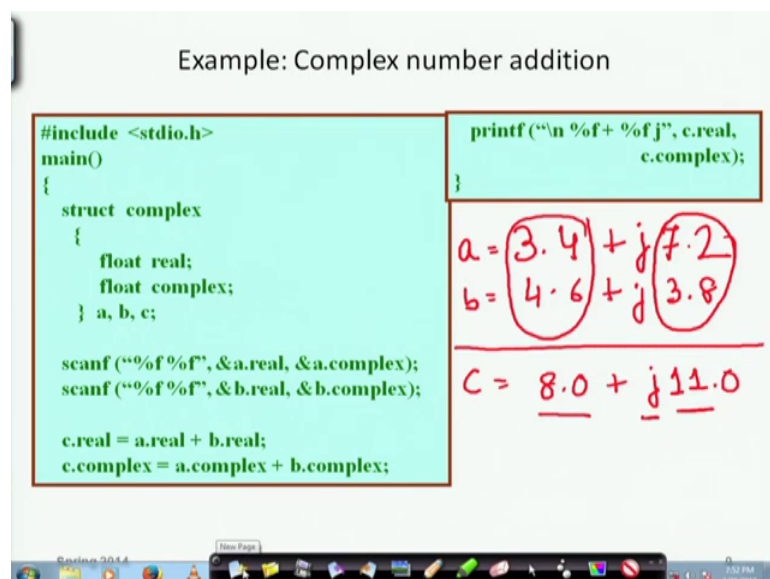
```
#include <stdio.h>
main()
{
    struct complex
    {
        float real;
        float complex;
    } a, b, c;

    scanf ("%f%f", &a.real, &a.complex);
    scanf ("%f%f", &b.real, &b.complex);

    c.real = a.real + b.real;
    c.complex = a.complex + b.complex;

    printf ("%f\n %f+ %fj", c.real,
            c.complex);
}
```

$$a = 3.4 + j7.2$$
$$b = 4.6 + j3.8$$

$$c = \underline{8.0} + j\underline{11.0}$$


So, what is being done is suppose I have got 2 numbers, 3.4 plus here I am using j some people I usually use i or j whatever you do? So, you write 7.2 that is a is real part is this, and b is real part is say 4.6 plus j 3.8 now if I add them c will be these real parts are being added. So, it is 8.0 plus the imaginary part is added that is 11 j 11 all right. So, what will be printed is 8 plus j then the complex part 7.0.

So, this is a I think this is clear to you this not very difficult to understand let us move ahead.

(Refer Slide Time: 08:22)

Example: Complex number addition

```
#include <stdio.h>
main()
{
    struct complex
    {
        float real;
        float complex;
    } a, b, c;

    scanf ("%f %f", &a.real, &a.complex);
    scanf ("%f %f", &b.real, &b.complex);

    c.real = a.real + b.real;
    c.complex = a.complex + b.complex;

    printf ("\n %f+ %fj", c.real,
            c.complex);
}
```

Scope restricted within main()

Structure definition And Variable Declaration

Reading a member variable

Accessing members

Spring 2011 9

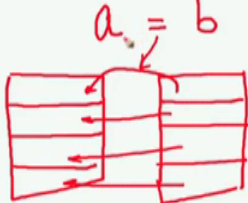
So, you have seen here, if you note here you have got the structure definition specified and here are the very variable definitions. And here we are doing the reading and adding them part by part how am I accessing the members. I am accessing the members using this dot operator and whenever I am reading the member I am actually reading the members, I am not reading the structure as a whole.

Now, one thing to note is when I am declaring this structure inside the main, then the scope of these variables real and complex etcetera or this structure as such is its scope is within the main. So, if a structure is defined within a function, then as we exit that function the life of that particular structure will end at that point.

(Refer Slide Time: 09:24)

Comparison of Structure Variables

- Unlike arrays, group operations can be performed with structure variables.
 - A structure variable can be directly assigned to another structure variable of the same type.



Windows taskbar at the bottom shows the date as 20/11/2014 and the time as 7:04 PM.


Now, there are some things which we can do in a much more simpler way, you remember that in the case of arrays we could not compare arrays together. That array a 1 is equal to a 2 that was not possible, if we had to compare the array I had to check them element by element. On the other hand in the case of a structure, a structure variable can be directly assigned to another structure variable of the same type. For example, here if a is a structure, then I can assign another structure b to it directly.

So, all the elements if they are of course, of the same type; So, here is one structure, here is another structure now if I do this operation ; that means, assignment then this will be copied here, this will be copied here, this will be copied here, this will be copied here all right.

(Refer Slide Time: 10:36)

Comparison of Structure Variables

- Unlike arrays, group operations can be performed with structure variables.
 - A structure variable can be directly assigned to another structure variable of the same type.



Spring 2014 7:05 PM 1/20/2014

But if a and b were arrays in that case, that would not have been possible that a is an array and b is an array. So, the elements just by assigning a assign b, I could not assign these b elements to a that is not allowed in the case of an array; however, that is possible in the case of a structure.

Secondly now here one important thing is that they must be of the same type all right otherwise of course, the members and the fields will not match.

(Refer Slide Time: 11:11)

Comparison of Structure Variables

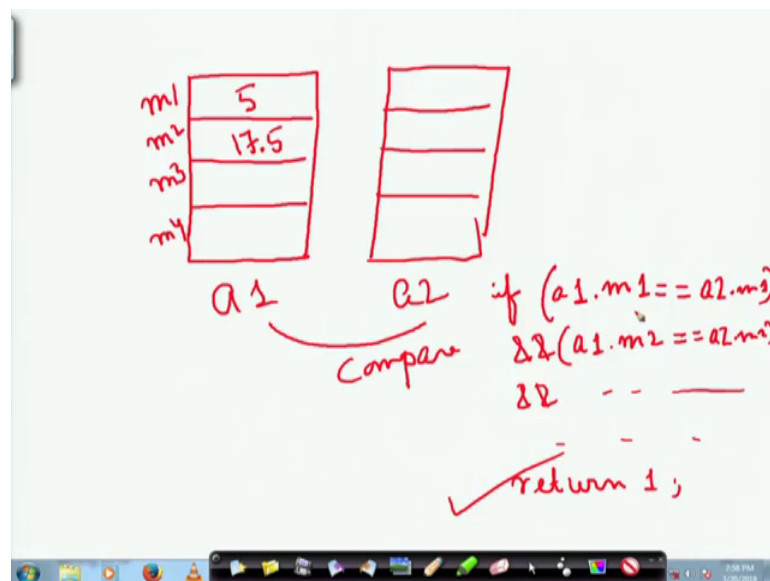
- Unlike arrays, group operations can be performed with structure variables.
 - A structure variable can be directly assigned to another structure variable of the same type.
 - `a1 = a2;`
 - All the individual members get assigned.
 - Two structure variables can not be compared for equality or inequality.
 - `if (a1 == a2)`
 - Compare all members and return 1 if they are equal; 0 otherwise.

Spring 2014 10

So, a 1 assigned a 2 I can do, all the individual members get assigned. Two structure variables cannot be compared for equality or inequality. Now this is important that two structure variables I cannot compare them, that structure a 1 whether that is equal to a 2 that will have to do compare I mean, what I said just a couple of minutes back was inadvertent that is wrong.

Structures we cannot compare we can assign in a short, but comparison of structures we cannot do as a whole all right. Two structure variables cannot be compared for equality or inequality now for that, what we have to do we can write a simple function say I can write and you can take it as an assignment, that we can write struct comp, struct c m p where you will take 2 structures as input and we will compare them and how do you compare them? We will do say variable a 1 dot member 1 equals to is whether its if it is equal to variable a 2 dot member 1, in that way if you do then it is possible.

(Refer Slide Time: 12:32)



So, what we can do here is, I have got two structures and there are 2 variables of the same type of the same structure type one is a 1 and one is a 2 and I have got some values here all right. And say this is member 1, member 2, member 3 member 4 like that now in order to compare if I want to compare them what I have to do?.

I will have to see if this is a 1, a 1 dot m 1 is this also m 1 is equal to a 2 dot m 1 and a 1 dot m 2 is equal to a 2 dot m 2 and so and so forth. If I do then return 1 that could be my function ; that means, if all these are matching, then only I will say that they have been

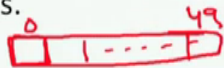
compared to be the same otherwise it will return false. So, here other else return false. So, you have to write it nicely you can take it up take it up as an assignment arrays of structures.

(Refer Slide Time: 14:07)


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[5].roll_number

Spring 2014

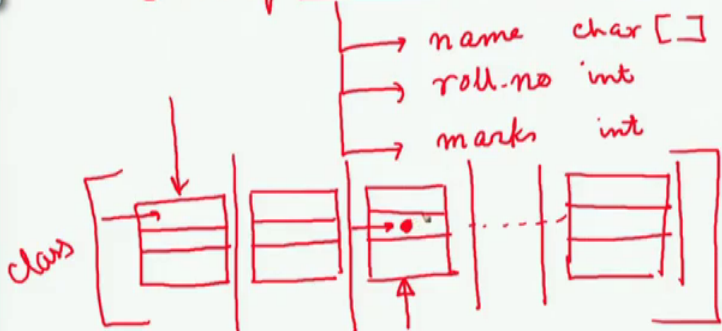


Now, once we know this, we can now make an array of structure why is that useful let us take one problem. Let us say that I am trying to store the information of a class of students.


(Refer Slide Time: 14:24)

Class of students 40 students

- name char []
- roll-no int
- marks int



class [0]. name
class [2]. roll-no



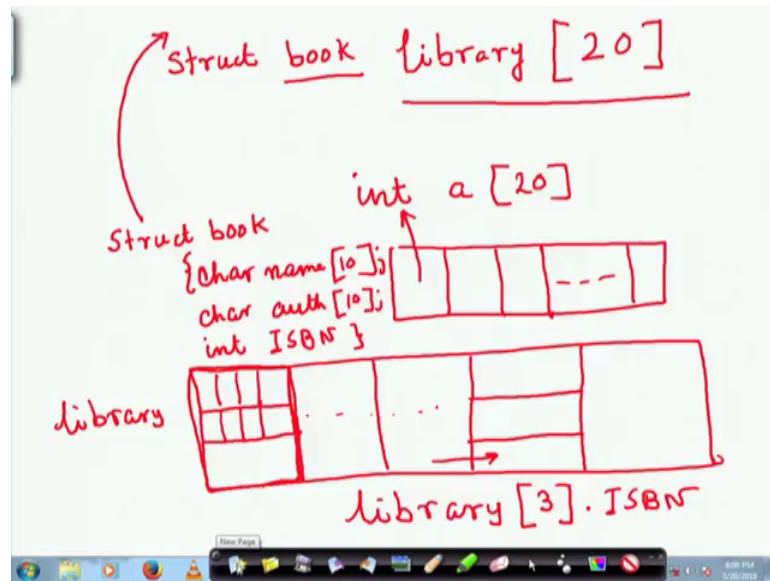
Where maybe there are say 40 students and their information about each student has got different components; name in a particular class, roll number and say marks total marks. So, marks is an integer let me say, roll number is also an integer and name is a character array something all right. Now that so, this box this can be also depicted as a box having three fields name, roll number and marks, but I want to have that for all the 40 students.

So, I will have 40 such small boxes, each having 3 fields one after another. So, that can be represented as an array you know that in an array we can store data of only the same type. So, that is not causing any problem here, because each of these boxes are of the type struct students therefore, I can very well consider them to be an array of such structures.

So, if I call this class is this array, then each of these boxes are an element of that array. So, if I say class 0, which element am I looking at? This element right; if I take class 0 dot name, then which field am I looking at? This particular field; if I write class 2 dot roll number, which field am I looking at? Class 2. So, it is an array 0 1 2 you know that just in c any array starts with 0. So, 0 1 2 I come over here and which field am I looking at roll number roll number is the second field. In that way I can access each element of this array, either as a structure or as an individual member of that structure all right.

So, in such cases or the same thing could be applicable, if I want to represent a library. So, in that case what will happen? A library will consist of books right. So, suppose it is a small library, where I have got a library of 20 books all right 20 books.

(Refer Slide Time: 17:50)



So, books is an array of type what or let me let me name it different way. So, I am say in my library is consisting of 20 books at most, and what is the type of library when we write in a array? We write int a 20 float a 20 like that. So, here what is the type of library? Type of library will be struct book. So, because every element of this library, when I write int a 20; that means, I am talking of an array where there are 20 elements and each element is of type int.

So, when I am talking of this struct book library 20 what will that mean? That will mean that here is an array library having 20 such field 20 such positions elements and each element is a structure called book. I have not defined here the structure of book. So, I should have done it earlier.

Suppose I do here struct book, char name 10 semicolon that is the name of the book. Char author 10 the author of the book int ISBN and I close that. Suppose this is a structure of the book having 3 fields only I have it could have been more fields number of pages year of publication etcetera I am not showing that. So, each of these elements have got 3 fields here name which is itself an array of 10 characters, author which is itself an array of 10 characters and an integer ISBN number.

So, this is the whole element, the first or 1 element of the array library. And library is an array which houses 20 at the most such individual structures. So, if I say library 3 dot

ISBN where am I going which one I am referring to? 0, 1, 2, 3 library 3 and the last element of that this is the ISBN number because this is a last one all right.

So, therefore, using structure allows us to store different types of data together, and again array of structure is a very powerful facility that has been given to us, using which we can store many more things and we get much more flexibility. Let us go here, once a structure has been defined like the structure book that was defined we can declared an array of structures, for example, struct student class 50.

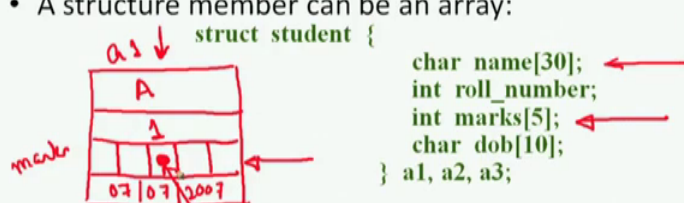
So, note here that class is an array having 50 elements. So, it is something like this here is class with 50 elements 0 to 49, and what is each element? Each of these elements is a structure of type student must have been defined somewhere earlier.


(Refer Slide Time: 22:47)

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]`



The individual members can be accessed by class whichever element are you want to have say for example, struct student, one field is int roll number, there is a name, there is marks and character. So, this is the student and I have got 3 variables a 1, a 2, a 3. The array element within the structure and I have defined an array earlier that is a class of 50 right. So, now, I have got an array of which every element is a structure like this, and the structures are variables a 1, these are also the student variables also my class is consisting of the same structure.

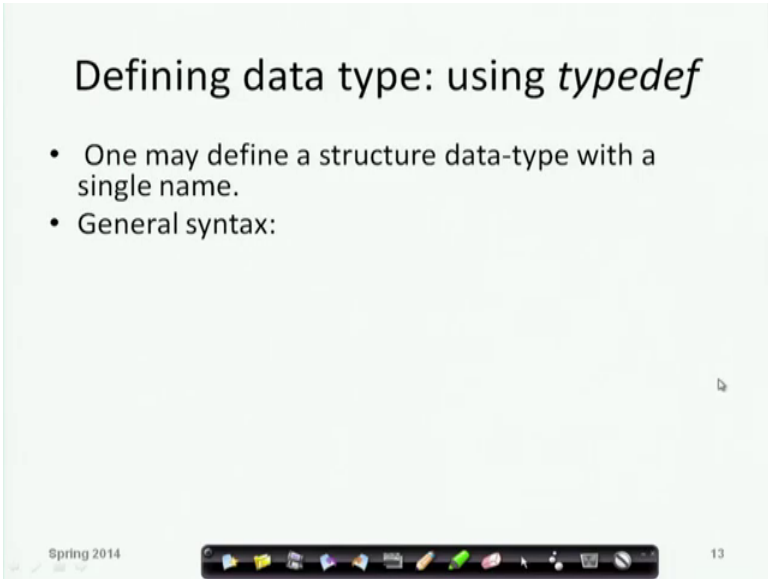
The array element within the structure can be accessed as now you can very easily guess, if a 1 dot marks 2. Now, here this example is telling us something more, here this example is showing that any element of have and a member of this structure student can also be an array, that we had defined earlier this is also an array.

Now, suppose I have got this student a 1 his name is a, roll number is say 1, and marks is itself an array where the marks of literature, history geography, maths, science everything is stored and the date of birth is a character array all right. Now so, it is say 07 07 2007. So, that is a character array.

Now, when I am saying a 1 marks 2; that means, I am going to this field is the marks field, I am going to the a 1 structure a 1 variable, and I am taking its structure and coming to the marks field or membership member and I am coming to the element 0 1 2 this particular element this particular element I am coming to by this, this you must understand.

So, I can have an array within a structure, also I can have an array of structure. Till now what we are discussing was and here what we discussed was an array of structure right. And here what we have just now shown you is what we have just now shown you is an example of that an array can also be a part of a structure.

(Refer Slide Time: 26:02)



The image shows a presentation slide with a light green background. The title is "Defining data type: using *typedef*". Below the title is a bulleted list with two items: "One may define a structure data-type with a single name." and "General syntax:". At the bottom of the slide, there is a Windows taskbar with various icons and the text "Spring 2014" on the left and "13" on the right.

Defining data type: using *typedef*

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

Spring 2014 13

Next lecture we will be talking about a new thing that is that is also very important type def. That is very much useful for that facilitates us to create new type definitions and we can write much simpler programs with this we will take it up in the next lecture.