

**Programming, Data Structures and Algorithms**  
**Prof. Shankar Balachandran**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

**Module -11A**

**Lecture - 36**

**What is a structure?**

**Example: Points in a plane**

**Accessing members, More examples**

**Nested structures: point, line, triangle, rectangle**

**Defining new data types: typedef**

Hello, welcome back all of you. I hope you have been enjoying the lecture. So, far and you have been putting in some time to work on the home work exercises as well. So, in this lecture we are going to look at what are called structures. So, we looked at one kind of aggregate data type called arrays earlier and in this class we are going look at what are called structures.

(Refer Slide Time: 00:35)

**Structures**

- Collection of one or more variables, possibly of different types, grouped together under a single name for easy handling.
- For example- a structure which represents a point in a two dimensional plane

```
struct point{  
    int x;  
    int y;  
};
```

A mechanism for defining compound data types

By itself it reserves no storage

So, structures as arrays are a collection of one or more variables. So, you have one or more variables,, but as opposed to arrays structures can have different types group together. So, if we look at an array right it is an aggregate type of the same data type. So, I can have an array of integers or array of floating point numbers and so, on, where as structure let us you aggregate data which are even of different data types.

So, let us take for example, point in a two dimensional plane, a two dimensional plane will have an x coordinate and a y coordinate. And you want to keep this group together,

because that is what a point is and one way to do that is as follows. So, you have a something called a point which we call a struct. So, struct is a keyword in C, we say that point is a data type which has two things, some integer called x and another integer called y from now on point can be treated us a new data type that you created in your program.

So, this is something that is absolutely new for your setup so, far. So, far we have been using all the data types that are already in place, like int and float and character and so, on,, but for the first time we are seeing, how to create your own data types. So, we have created a new data type called point which has x and y has two members inside it. So, by doing this you get a mechanism for defining compound data types as of now, you do not have a storage this is like saying I have an integer. So, only we need do something like int a comma int b and so, on, you have variables of the names a and b just by having int in it you do not have a storage space. Similarly, when we have the basic data type called struct point, we are saying that this structure is going to have two things an integer called x and an integer called y, that is group together. And this collection is useful instead of looking at as a two different integers, this collection is going to be for a representing a point.

(Refer Slide Time: 02:47)

Point in 2D → 2 integers

- Different ways of declaring structure variables

```
struct point{  
    int x;  
    int y;  
} point1, point2;
```

```
struct point point1, point2;
```

```
struct point point1 = { 3 , 2 };
```

Storage is allocated now.

You can also initialize on declaration.

So, let us look at this a point in 2D is now two integers, there are different ways to declare structure variables. So, let us look at this top part, you have a struct point int x int y and we have these braces closing it followed by point 1 and point 2. So, the way it interpret this is point 1 and point 2 are two variables of the data type called struct point

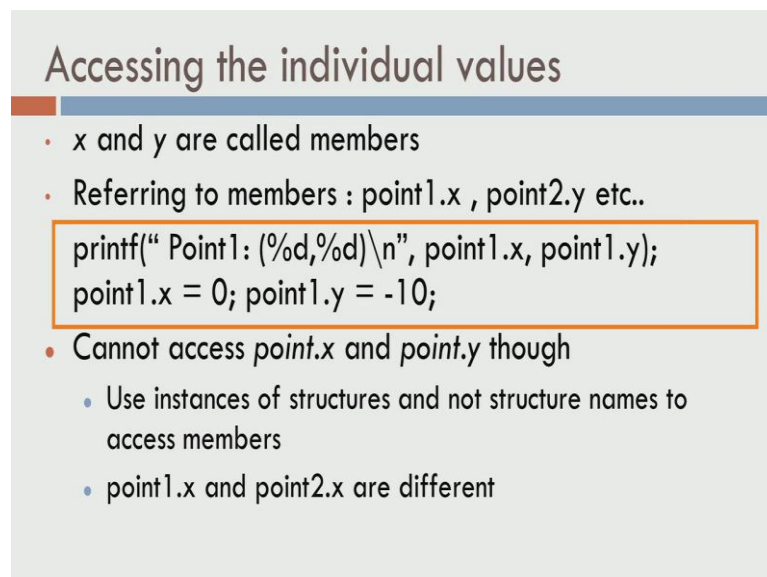
and the struct point data type has two integers in it x and y. So, this is one way to do it.

Or if you have already defined only the structure without doing declarations for the variables, we can declare only this part saying that my point is a data type containing two integers, you could leave it at that and then come back and say that I want two variables, point 1 and point 2 they are both of the types struct point. So, if you look at this line it is quiet similar to what we have done for integers and floats and so, on.

So, on the left side you start with the data type and then you have a comma separated list of variable names, only that the data type has, is keyword struct in front of it and in name that we have assign to the data type. So, at this point the storage for point 1 and point 2 are allocated. So, point 1 will have two integers and point 2 will also have two integers, there is one way to initialize this, that is in the last line here struct point point 1 is 3 comma 2. So, at this point you are saying that point 1 is of data type struct point.

And since there are two members x and y, the two members will get the values 3 and 2 respectively. So, that is one way to say that point 1 should have it is x as 3 and y as 2. So, this is also very similar to what we have for basic data types, where we say int x equal to 5 for instance would mean, you are not only declaring a variable called x, you also have the storage declare for it plus the initialization of the value to 5.

(Refer Slide Time: 04:56)



### Accessing the individual values

- x and y are called members
- Referring to members : point1.x , point2.y etc..

```
printf(" Point1: (%d,%d)\n", point1.x, point1.y);  
point1.x = 0; point1.y = -10;
```

- Cannot access *point.x* and *point.y* though
  - Use instances of structures and not structure names to access members
  - point1.x and point2.x are different

So, let us see how to access the individual elements x and y are called members and referring to the members is done with what is called the dot operator or the period. So, point 1 is a variable, point 1 dot x will give you the x coordinate and point 2 dot y will

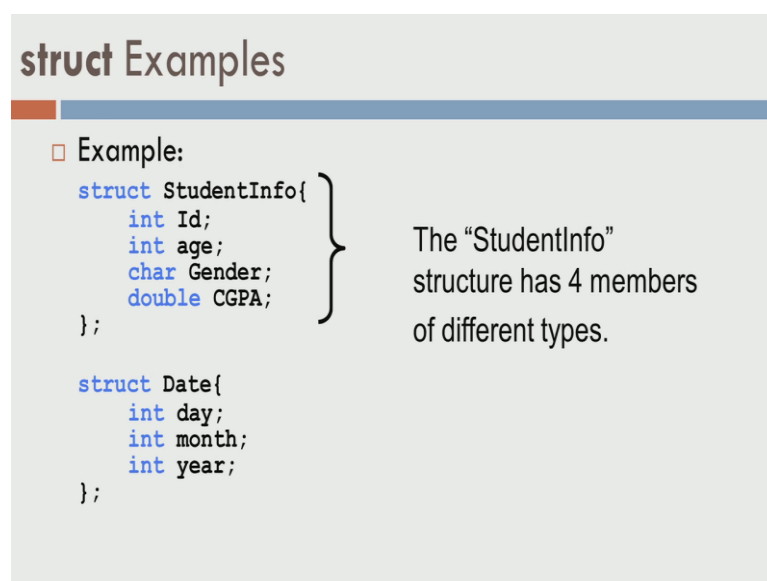
give you the y coordinate of point 2 and so, on. So, let us see this small piece of code, let us say I want to print point 1 on the screen. So, printf point 1 percentage d percentage d print point 1 dot x and point 1 dot y.

So, this statement will look at point 1 dot x the x member of point 1 and print it as an integer and look at the y member of point 1 and print it as an integer. So, that is what this line does, it is not just you can use them for printing. So, you have read the values of point 1 and point y, you can also go and change the contents of point 1, you can also write to it. So, point 1 dot x equals 0, point 1 dot y equals minus 10.

So, this one changes the value that you currently have two a point called 0 comma minus 10. So, it could have been something else before and now you could change it to 0 comma minus 10. So, one small thing that you have to watch out for is that, you cannot access point dot x and point dot y, remember point is a data type point, the data type does not mean that you already have storage, only when you have a variable of the certain data type you are allocated storage.

So, you can only use instances of the structures, namely the variables it you have used and not the original data type itself. So, point 1 dot x is point dot x is not point 2 dot y is point dot y is not, it should also know that point 1 dot x and point 2 dot x are actually two different variables, they have two separate memory locations and they do not get mixed up.

(Refer Slide Time: 06:56)



### struct Examples

- Example:

```
struct StudentInfo{
    int Id;
    int age;
    char Gender;
    double CGPA;
};

struct Date{
    int day;
    int month;
    int year;
};
```

The "StudentInfo" structure has 4 members of different types.

So, let us see various other examples of structures. So, one classical example is of a

student who has a student id and as an educational instantiation I want to track, what is the age of the student is the student male or female and what is the student's CGPA and so, on. So, this is one logical group of things, I have the student id and along with that I have an integer called age, I have a character I would probably put m or f depending on whether the student is male or female and the CGPA is usually a floating point number. So, I am going to keep it as a double CGPA.

So, this is the logical collection of things, instead of keeping them as four separate integers if I have it as a structure. So, this collection has a meaning. So, it is all the information about a student and we call this information here student info. Similarly, if I am going to look at a date, I have three things that make a date, the day, the month and the year, we need all these three.

Again I could have kept it as three separate integers, but putting all of them as one logical unit makes sense. Because, then I can look at it today this date or setup today's date to be this and so, on, instead of dealing with three separate integers which have no relationship to each other.

(Refer Slide Time: 08:13)

### struct examples

□ Example:

```
struct BankAccount{
    char Name[15];
    int AccountNo;
    double balance;
    struct Date Birthday;
};
```

The "BankAccount" structure has simple, array and structure types as members.

Finally, let us look at another example called bank account. So, this is supposed to have details of a bank account. I of course, have the name of the person who has the bank account, in this case you have name which is a character array of 15 bytes. So, the name can be up to 15 bytes, then there is an integer account number in this case account number is expected to be integral. And the balance that you have in the account is

suppose to be a double value and am I want to track the birthday of my customers and that in turn is a structure.

So, this is what I was talking about earlier that you could actually mix and match data types of different kinds and through them into a structure. So, in this one we have a character array of size 15 and integer, a double and within a structure we have another structure called date or birthday. So, there is a member called birthday whose data type is struct date. So, date I already mention has three members. So, in some sense we have a nested structures here. So, we have a structure called structure date within this structure called bank account.

(Refer Slide Time: 09:25)

### A rectangle

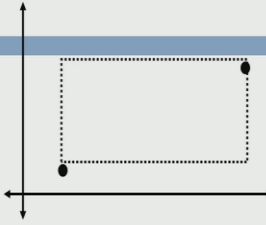
```
struct rectangle{  
    struct point pt1;  
    struct point pt2;  
} rect1;
```

- Accessing points in the rectangle

```
rect1.pt1.x = 4;  
rect1.pt1.y = 5;
```

Or

```
rect1.pt1 = { 4, 5};
```



x is a member of pt1 which is a member of rectangle.

So, this kind of nesting is really useful. So, let see another example here let say we have a rectangle and I want the rectangle to be a data type by itself,, but the rectangle is specified by the left bottom point and the right top point. So, I have two different points that define the rectangle, namely the left bottom point and the right top point and rectangle itself is a data type. So, this is useful for example, I want to go and draw the rectangle I will pass this structure called rectangle to the draw function if.

So, let us look at this struct rectangle. So, it says that we want a new data type called rectangle and this in turn consist of two members, namely pt1 and pt2, pt1 is struct point data type and pt2 is also struct point data type. And somewhere we have to remember and know that pt1 stands for left bottom corner and pt2 stands for right top corner. Now, if we want to access something in this nested structure you could do this.

So, rect1 is a variable of type rectangle, you can see the declaration here the data type is in the top and rect1 is the variable of the type rectangle. So, rect1 dot pt1 dot x refers to the variable rect1 it has two members pt1 and pt2 and the pt1 member has x as one of its member. So, you are looking at member of a member and you want that to be set to 4 and rect1 dot pt1 dot y to be set to 5. So, this sets of the rectangle to have the left bottom point as 4 comma 5. You could also do this, rect1 dot pt1 is 4 comma 5, we already saw this example, only that we have a nested structure case. So, rect1 dot pt1 the x member takes the value 4 and the y member takes the value 5.

(Refer Slide Time: 11:24)

## Nested Structures

- We can nest structures inside structures.
- Examples:

```

struct point{
    int x, y;
};
point P;

struct line{
    point p1, p2;
};
line L;

struct triangle{
    point p1, p2, p3;
};
triangle T;
    
```

The diagram illustrates nested structures on a 2D coordinate system. A point P is shown with coordinates (P.x, P.y). A line L is defined by two points: (L.p1.x, L.p1.y) and (L.p2.x, L.p2.y). A triangle T is defined by three points: (T.p1.x, T.p1.y), (T.p2.x, T.p2.y), and (T.p3.x, T.p3.y). The points are marked with red dots, and the line and triangle are drawn in blue.

This notion of nest nested structures is really useful to construct a lot of things, let us do this example one more time, similar nested structures. So, we have a struct point which has two members x and y and I have a variable p of the data type point, then let us say I have a struct line which takes two points. So, a line in a two dimensional plane is defined with respect to two points and I have these two points, this blue line here is line l and line l has p 1 and p 2.

So, this point here for line l which is p 1 is x coordinate is l dot p 1 dot x and the y coordinate is l dot p 1 dot y, this point here is p 2 of l it is x coordinate is l dot p 2 dot x and its y coordinate is l dot p 2 dot y. Similarly, I can make a triangle and I have to specify three points for it. So, the data type is called triangle and the variable is called t. So, t dot p 1 dot x and t dot p 1 dot y is one of the corners of the triangle and there are two other corners namely p 2 and p 3.

(Refer Slide Time: 12:44)

### Assigning Values to Variables Using The Picture Below

```
struct point P;  
struct line L;  
struct triangle T;  
P.x = 4;  
P.y = 11;  
  
L.p1.x = 2;   T.p1.x = 2;  
L.p1.y = 7;   T.p1.y = 0;  
L.p2.x = 10;  T.p2.x = 6;  
L.p2.y = 9;   T.p2.y = 5;  
T.p3.x = 8;  
T.p3.y = 3;
```

The image shows a coordinate plane with a grid. A point P is plotted at (4, 11). A line segment L is drawn between points (2, 7) and (10, 9). A triangle T is drawn with vertices at (2, 0), (6, 5), and (8, 3). The coordinates for each point are labeled in red text next to the points.

So, let say I want to setup point p to have 4 comma 11, line l 1 to have 2 comma 7 going to 10 comma 9 and triangle with these three as coordinates, how do we do it. So, this is how do it we first say that there is struct point p, we already have struct line l and struct triangle t, it is assume that these declarations for p l t are in place and now we have variables p l and t. So, to make this point p to be 4 comma 11, we can say p dot x is 4 and p dot y is 11 and for making the line 2 comma 7 to 10 comma 9 l dot p 1 dot x is 2 and l dot p 1 dot y is 7 takes care of this point l dot p 2 dot x equals 10 and l dot p 2 dot y equals 9 takes care of this point.

So, we have two points and for each point we are given the x coordinate and y coordinate. And finally, for the triangle will we need three points and for each point we give the x and y coordinate.



(Refer Slide Time: 13:55)

## Defining new types

- 'typedef' is used for creating new data types, for example

```
typedef int Age;  
Age myAge = 99;
```

- typedef and Structures – especially a good combination

```
typedef struct point pointType;  
pointType point1, point2;
```

- This is equivalent to struct point point1, point2;

So, this defining new data types is a very useful and powerful thing and sometimes it gets very tedious to say that it struct rectangle and struct pt and so, on and C programming language gives you this short cut called typedef. So, typedef is use to create new data types for example, let say I have an integer and the meaning that I want to attach the integer is age. So, you do not want to accidentally mix it with something which is of type let say volume or something which is of the type date and so, on.

So, this is age and it is an integer, if I just say it is int I could accidentally use this variables somewhere else. But then, now what I am going to do is, I am going to say age is typedef to be an integer. So, what I mean by that is, wherever I see age as a data type intern it is actually just an integer. Now, if I declare age my age equals 99 it is clear that my age is of age data type and I do not want to mix it with things like, volume or length and so, on, it is actually about age.

So, this combination of doing typedef is particularly useful in structures. So, for example, I could. So, we were using this struct point struct point and so, on repeatedly. Now, I am going to use a short cut, which says typedef struct point point type. So, what this does is it defines a new data type called point type, which is actually a struct of type point. So, it is a structure of type point and the nick name for that is point type.

From now on I can avoid saying struct point point 1 and so, on and instead I can say point type point 1, point 2. So, in the last line here we can see that this is very similar to what we have do for basic variables, we put the variables data type first and then a

comma separated list. Only the variables data type does not have this extra thing called struct followed by the structure name, instead it has this nick name called point type. So, this is actually equivalent to writing struct point point 1 and point 2. So, it avoids typing this struct point every time.