Introduction to Problem Solving and Programming
Prof. Deepak Gupta

Department of Computer Science Engineering

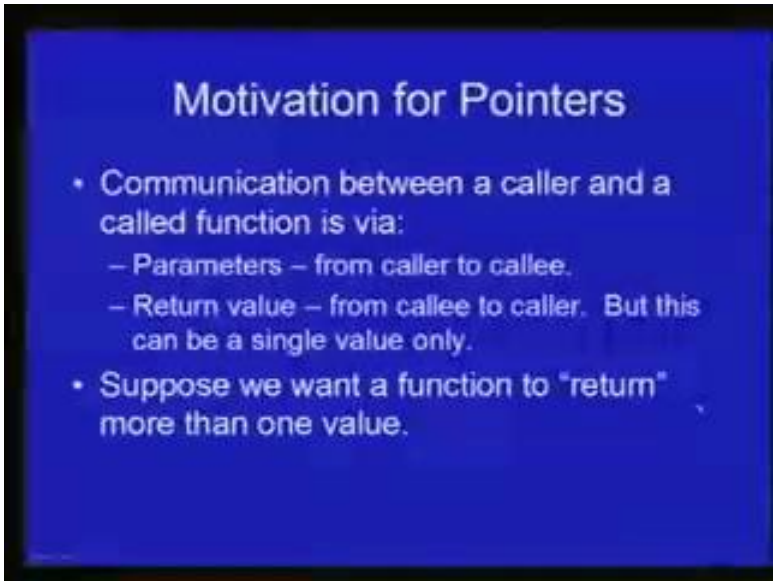
Indian Institute of Technology, Kanpur

Lecture -17

(Refer Slide Time: 00:26)

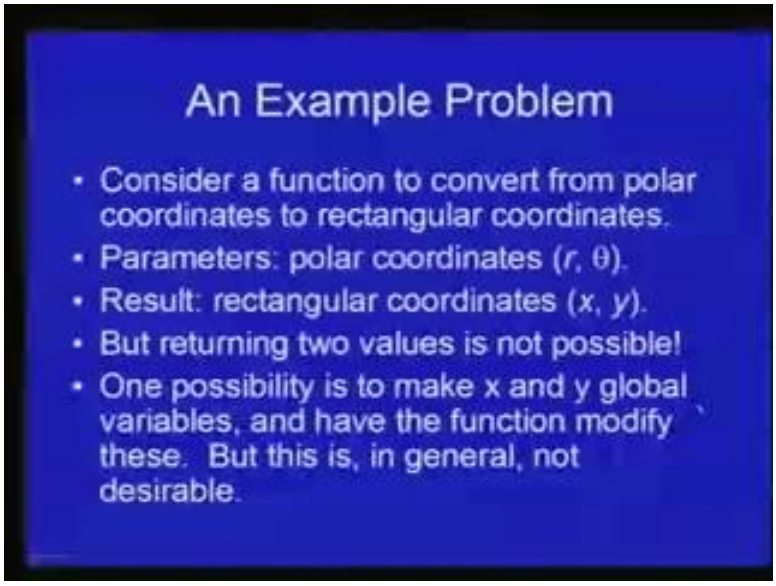
In today's lecture, we will look at another very important tool in programming that is the notion of pointers. So, let me try to motivate some use for pointers.

(Refer Slide Time: 00:36)



As we have seen when we are discussing functions the communication between the caller function and the calling function that is the function that has been called is in two ways by communication we mean the flow of information or the transfer of information. One is the caller can pass some argument to the callee function, this communication is from the caller to the callee; and the reverse direction callee function can return a value to the caller function in this communication is from the callee to the caller. But note that the return value as you have seen can be a single value, it cannot be more than one value. Now, suppose we want a function to return more than one value then how do we achieve that.

(Refer Slide Time: 01:19)



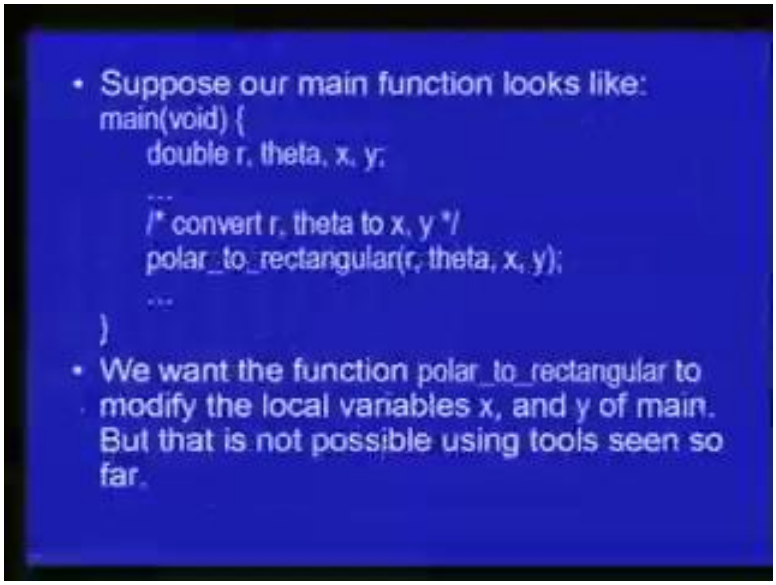
So, here is a example problem in which you might want to actually do that. Suppose we want to write a function to convert from polar coordinates to rectangular coordinates may be the program that we are writing is a graphics program and deals with geometric objects and so on. So, we want to write a function which converts from polar coordinates that is r theta to rectangular coordinates that is x y. Now, how do we write such a function the parameters of the functions are clear they would be the values of r and theta and the return value should comprise of x and y, but that is not possible because we cannot return two values from a function.

One possibility as we know is that we could make the variable x and y - global variables, and then the function which converts some polar to rectangular coordinates could modify

these global variables instead of returning any values, but in general that is not a good idea we should avoid using global variables as far as possible. Because when two function share global variable that is they both access the same set of global variables then this increases what is known as binding between the two functions. And what binding means is that the two functions cannot really be understood in isolation that is for understanding one of these functions you have to also understand what the other function is doing. So, that makes it more difficult for somebody to understand the program by reading it, because he cannot just focus on one function at a time and consider it independently of the rest of the functions.

So, we should use global variables when it is really required, but in most cases, we should avoid using global variables and variables which really are of used only one function should be made local to that particular function. So, the question remains how do we solve this problem. So, we will solve this problem using pointer. I will show you the solution towards the end of the lecture, but having motivated the use of pointers. Let us now look at what pointers are and what we can do with them.

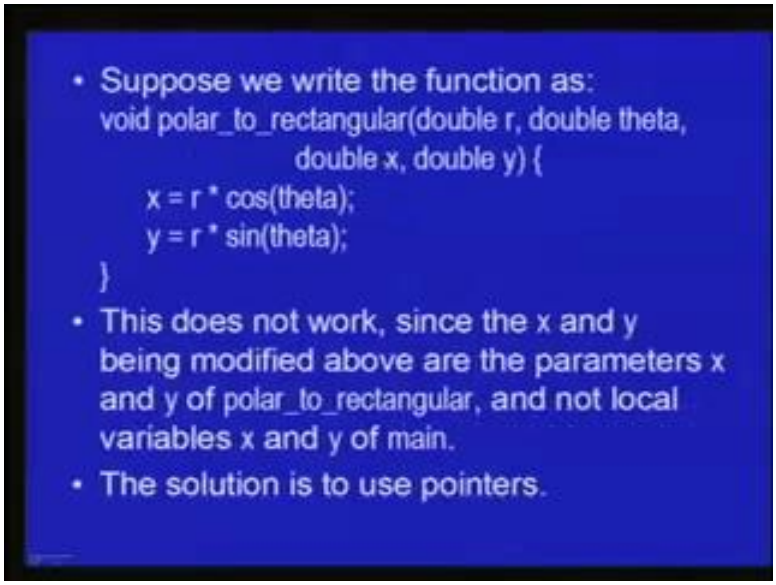
(Refer Slide Time: 03:28)



So, suppose we did not use pointers and wrote our polar to rectangular function by adding two extra arguments section y to it and the polar to rectangular function modified this x and y values. As you know what would happen is that when polar to rectangular modifies, this variables x and y what it is really modifying are the parameter of the

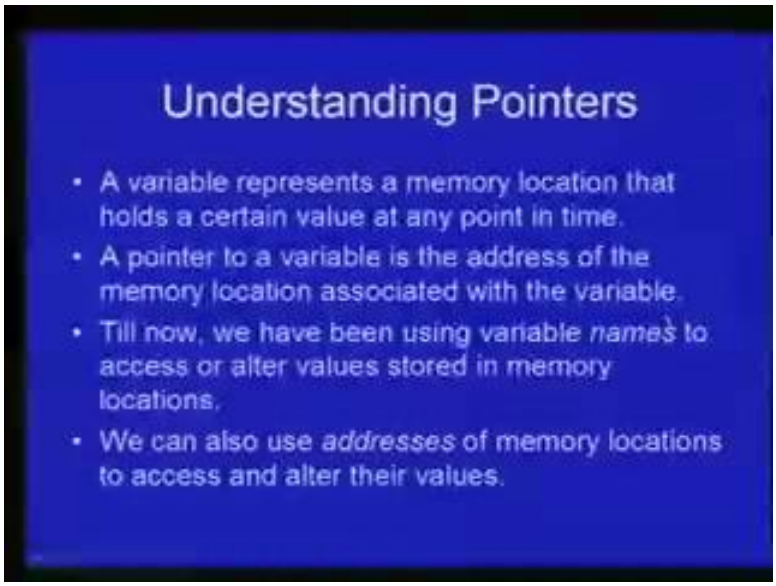
function polar to rectangular and this change would not have effect on these variables x and y of the main function. So, that is something that we can do only using pointers.

(Refer Slide Time: 04:11)



So, let us if we wrote the function like this as we have already seen, this does not work because the x and y in the calling function they do not really get modified. So, let us now try to understand what pointers are, but to understand that lets go back to our basics and first talk about what a variable is.

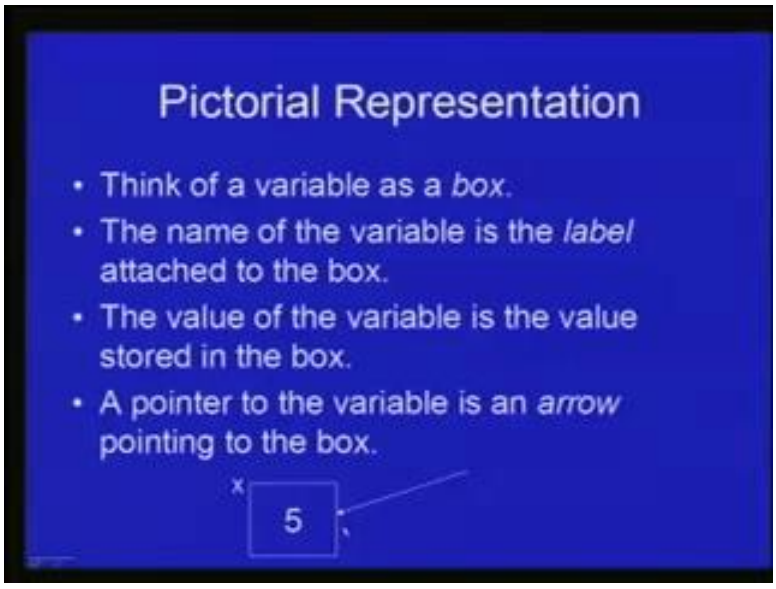
(Refer Slide Time: 05:50)



So, recall that a variable represents a cell or a location in the memory of the computer that holds a certain value at any given point in time. Recall that the memory consists of large number of memory locations or memory cells and each of these cells has a distinct address associated with it that is how at a lower level we identify or refer to individual cells in the memory. So, a pointer to a variable is nothing but the address of the memory location which is associated with that particular variable. So far, we have been using variables that is accessing their values and altering or modifying their values using only their names that is use the name of the variable to refer to it or to get its value or to modify its value and so on.

But using pointers what we can also do is to instead of using the name of a variable to access it, we could use its address to actually access the same variable, so that is what pointers allow us to do. It is actually not important to worry about the actual addresses of variables, because what the actual address of the variable is depends on what address the compiler really assigns to it. But the actual addresses are not important, what is more important to realize is that a particular value is the address of a particular variable. So, that is better understood using a pictorial representation.

(Refer Slide Time: 06:57)

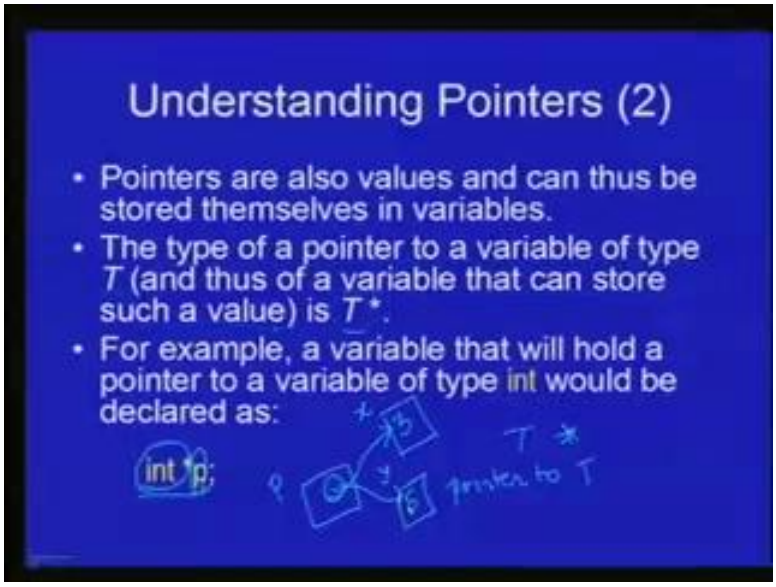


So, what we are going to do is to represent variables as boxes. So, each variable is a box in which some value can be stored or placed. The name of the variable we can think of as a label attached to that box. The value of the variable is the value stored in the box or

written in the box; and finally, we can represent or think about a pointer to the variable as an arrow pointing to the box. So, for example, here is a variable x with value five. So, we have represented this variable as a box, the value five is written inside it, and x is a label for this box, and a pointer to this box is nothing but we should think of it as an arrow pointing towards the box.

We will see that it is a very helpful to use this notation to understand what a particular program is doing, because we do not need to actually deal with actual numerical values of the addresses. So all the information that we need to know about a particular address or a pointer is that if the variable that it is pointing to. So, representing it as an arrow pointing towards the box representing a particular variable is a very powerful notation. And we will see that we can using this notation very easily visualize, what is going on in a program that is using pointers.

(Refer Slide Time: 08:48)



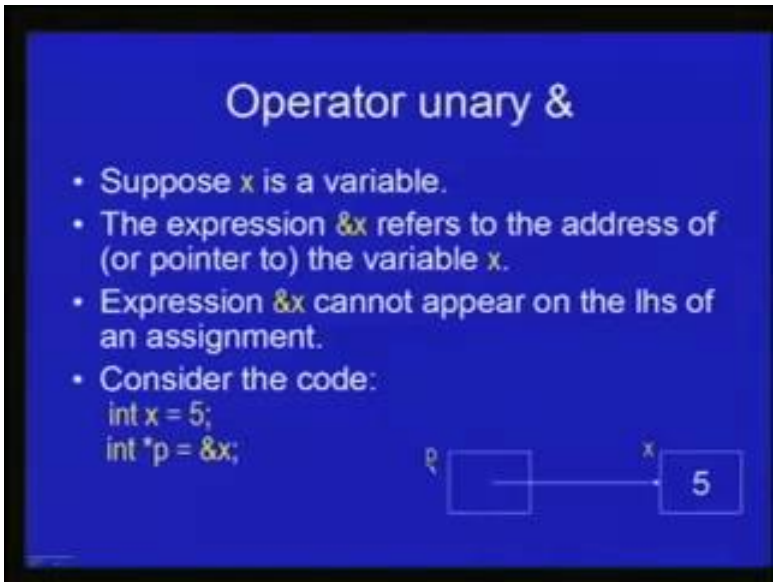
Now the next important thing to understand is that pointers are also values. And just like a values of types like integers and characters and floating points and and so on so forth. They can also be stored in variables and they can be passed as arguments to functions and so on so forth, but the type of a pointer is not int or float or char or something like this. There is a notation for specifying the types of pointer and the type of the pointer depends on the type of the variables that it can point to. A particular pointer variable can hold a pointer to only variables of a certain fixed type. For example, iif we say that we

have a variable which will hold pointers to integers then this variable can only hold pointers to a variable of the size int.

At different points in time in the execution of program, it may hold pointers to different integer variables, but it will always hold pointer to some variable, which is of type int. So, if a pointer is going to point to a variable of type T and therefore, also the type of the variable that is going to store such a pointer then the type of the pointer and type of the variable which will store such a pointer is T star, this is just a notation that we uses. So, T star should be read as pointer to T, and T itself could be any of types that we have seen it could be an int, it could be char, it could be a float or a double or whatever else.

As an example suppose we want to declare a variable that will hold a pointer to some variable of type int then this variable would be declared as this. Here p is the name of the pointer variable and int star is the type of this variable. So, what this declaration see says is that p is a variable which means that p is itself a box somewhere with the label p. And what this box will contain is not an integer, but a pointer or an arrow to some other box which contains an integer, which might be x at a certain point in time. At a certain later point in time we may modify the value of this variable and make it point to another variable may be y which itself has the value five and so on and so forth.

(Refer Slide Time: 10:23)



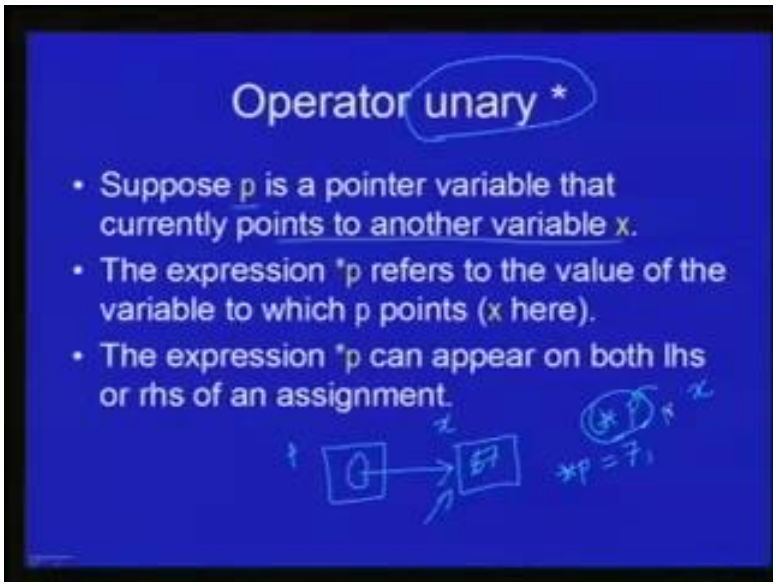
So, there are two operators associated with pointers. The first is the operator ampersand and the other is the operator star, both of which we will look at one by one. So, the

operator ampersand is a new operator, it should not be confused with another ampersand operator that exist in C. This is the unary operator, which means that it has just one operand. Now this operator can be applied to variable, it cannot be applied to an expression, which cannot appear on the left hand side of an assignment. So, suppose x is a variable then the expression ampersand x, refers to the address or a pointer to the variable x. So, using this particular operator, we can obtain the pointer to or the address of any variable that we care to, and this itself is value and it can be stored in a variable of an appropriate type and so on and so forth.

Note that this kind of an expression ampersand x cannot appear on the left hand side of an assignment, because we cannot modify the address of a variable. The address of the variable is fixed, and it is assigned by a compiler we cannot really change that. And therefore, this expression can appear on the right hand side of an assignment, because you can store it somewhere, but we cannot place it on the left hand side of an assignment because you cannot modify this value. So, as an example to illustrate the pictorial representation that we have been talking about consider this simple piece of code.

So, we have two variable declarations here. The variable x is of the type integer and has the value five; and the variable p has the type int star, which means that it will hold a pointer to an integer variable, and it is initialized to the value ampersand x. So, what this means really is that x has the value five, x is a box with value five in it; and p is another box labeled p, and what it contains is a pointer to or an arrow pointing towards the variable x.

(Refer Slide Time: 12:43)



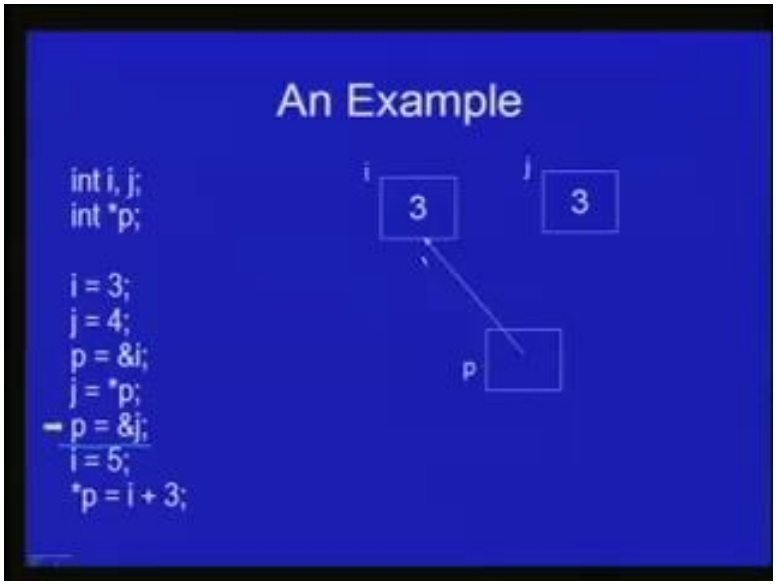
The other operator that is associated with pointers is the unary star operator and again please do not confuse it with the multiplication operator of C. The multiplication operator is a binary operator, whereas this is a unary operator and this is not an arithmetic operator, this does not really do any arithmetic. So, again let us assume that p is a pointer variable that currently points to the variable x. So, the situation is something like this. The type of p is some t star and the type of x is the type t, t could be some int or float or anything else.

Now, the expression star p refers to not the value of the variable p, but the value of the variable to which p points. So, the the expression p itself would be a pointer to the variable x, but the expression star p is the same as the value of the variable x of to which the variable p points. So, if x happened to have the value five and p happen to point to x then star p would have the value five also. So, this can be used on the right hand side of an expression to obtain the value of the variable to which a particular pointer is pointing. It could also be used on the left hand side of an of an assignment to modify the value of the variable to which the pointer is pointing.

So, for example, if we have the assignment star p assigned seven then this does not change the value of the variable p itself. What it changes is the value of the variable to which p is pointing which in this case is x and that value becomes seven. So, you can see in this case what is happening is that by using star p, we are actually referring to the

variable x itself, but instead of referring to the variable x by its name we are referring to the variable x by its address which is the value of the expression p.

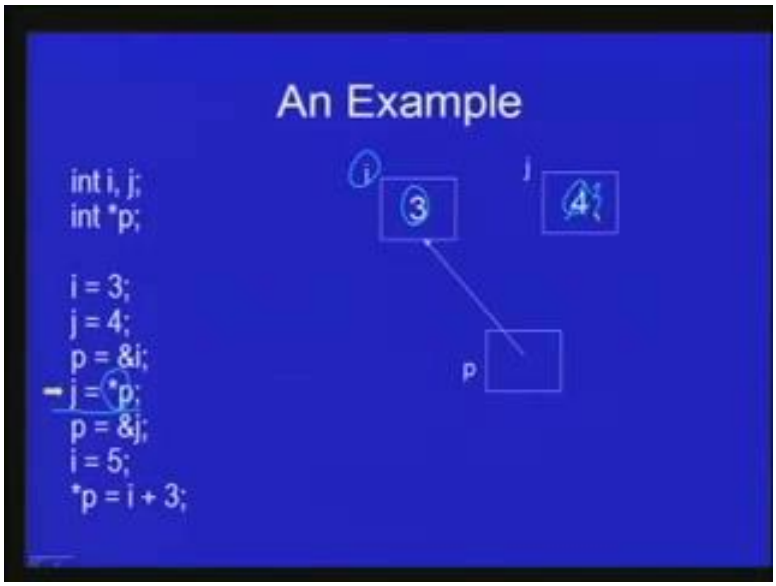
(Refer Slide Time: 15:08)



So, let us make this more concrete with the help of a detailed example. So, here is a piece of code and we will trace through this piece of code, and see exactly what is happening here. So, we have three variable declaration i, j and p and these are the three boxes for these three variables i, j and p; i and j are of type int, so these boxes will contain some integer values and p is of type int star, which means that p will hold a pointer to some integer variable. We have only two integer variables in this particular example, so p can hold a pointer to either i or to j.

Note that as we will see in this example in deed that at different points in time of the execution of this program, the variable p may hold pointers to different variable. So, let us just step through this code and see what happens. So, this is the statement that we need to execute.

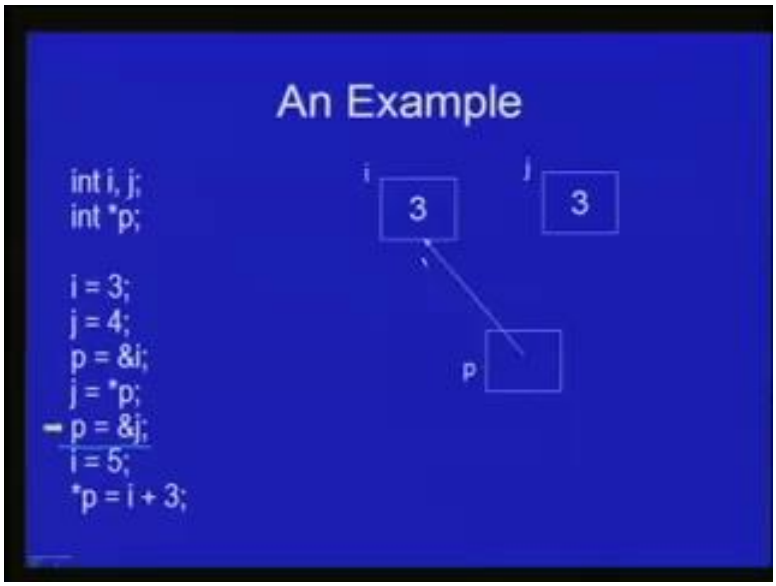
(Refer Slide Time: 16:05)



Next i assigned three when that is executed you all know what going to happen the variable i will get the value three, and control goes to next statement j assigned four. And when j assigned four is executed the value of j becomes four. So, till now everything is familiar. Now the next statement is p assigned ampersand j ampersand i. So, what that is going to do is the following, ampersand i represents a pointer to i or the address of i. So, this value is going to be stored in the variable p.

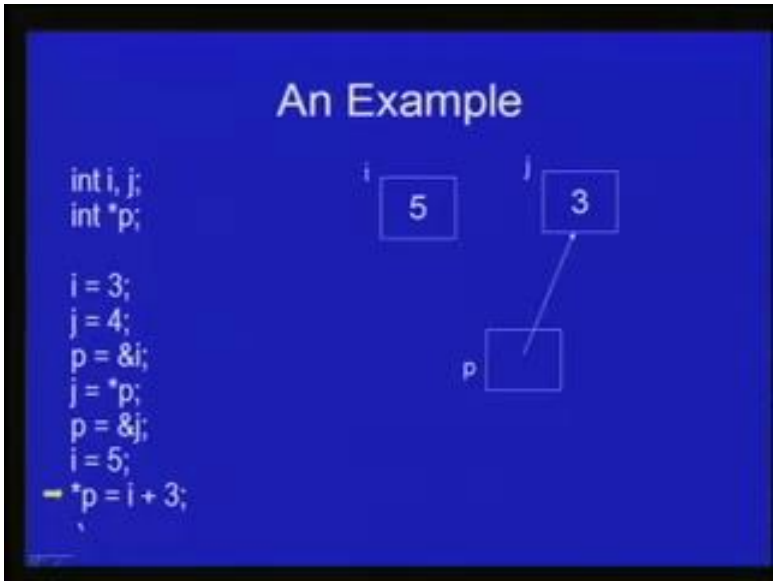
So, what that is going to do is establish an arrow from p to i and that is precisely what it does indeed to. And the next statement is j assigned star p. So, what should happen here, what is p, so the left hand side of the assignment is j, so clearly the value of j is going to change, but what is going to the new value, the new value is the value of the expression star p. The value of star p as we just saw is the value of the variable to which the pointer variable p points. Now p is pointing to the variable i and so the value of the expression star p will be three. So, what is going to happen is that j should become three when we execute this particular assignment statement as in deed that is precisely what happens.

(Refer Slide Time: 17:19)



Now, the next statement again modifies the value of p to the address of j what that means is that now p will stop pointing to variable i, this pointer will be removed and instead the value of p itself will change and it will start pointing to the variable j. So that is what is the effect of the statement.

(Refer Slide Time: 17:52)

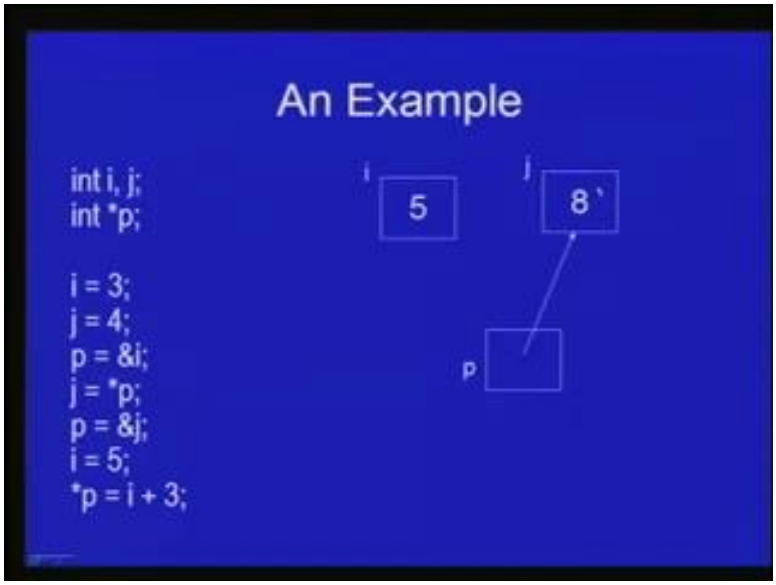


The next statement of course, is simple i assigned five. So, that just changes the value of the variable five; and finally, here we have an assignment in which the expression star p appears on the left side of an expression of an assignment. So, in this case what is going

to change, what is going to change is the value of the variable to which p points. Now at this point in time p is pointing to j. Note that the fact that the p earlier pointed to i is no longer relevant that fact is forgotten.

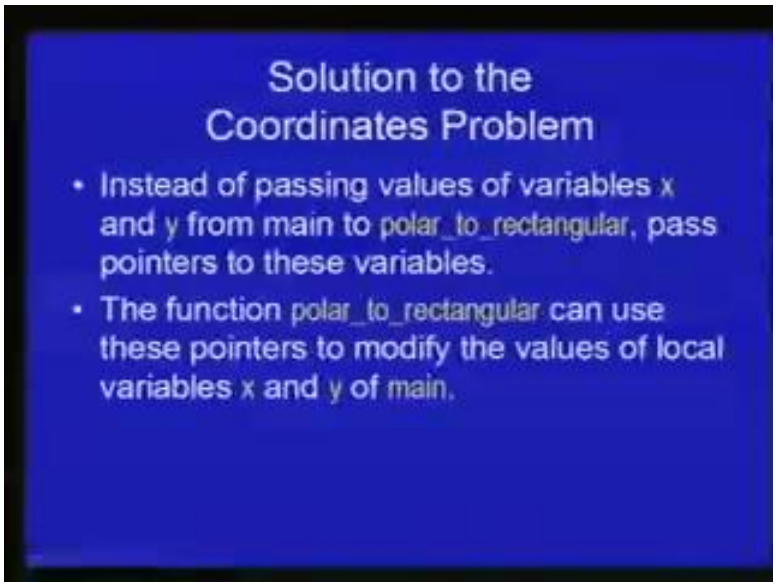
P is now pointing to j and therefore, when star p is assigned something the value of the variable j will change, and what will the value of variable j will become this will be the value of the expression on the right hand side of the assignment which is i plus three which is of course, is eight. So, this value should become eight as indeed that is what happens.

(Refer Slide Time: 18:40)



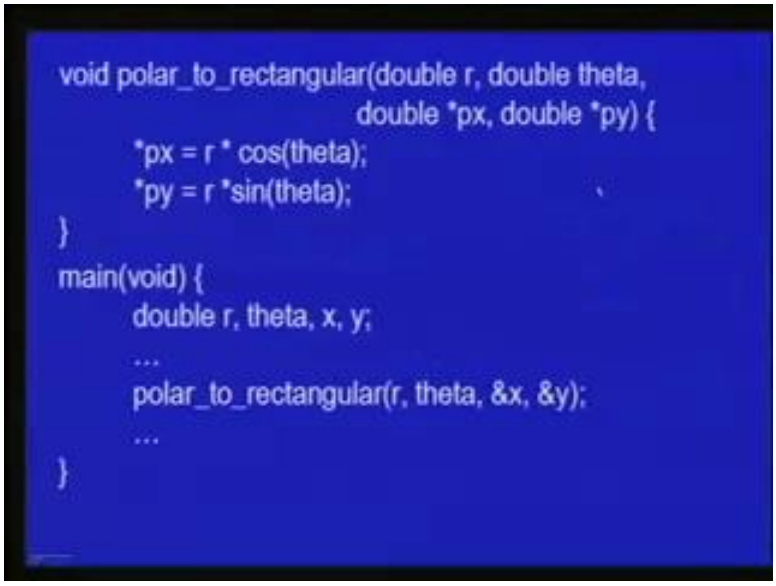
So, having understood the basic concepts of pointers, we can now solve the problem of conversion from polar to rectangular coordinates, and this example will also illustrate how we can pass pointers as arguments to functions. And so recall that the basic problem was that we wanted the function polar to rectangular to modify the variables x and y of the function main or it cannot access this variables of x and y of main by name because the scope of this variable is only within the body of the function main.

(Refer Slide Time: 19:29)



But it can access them if it has the addresses of these variables and how does it obtain the addresses of these variable while if the main function itself passes them as arguments to the function polar to rectangular. And the function polar to rectangular will have access to the addresses of these variables, and when it uses these address to access these variables that process is called dereferencing then it will be able to modify the variables x and y of main itself.

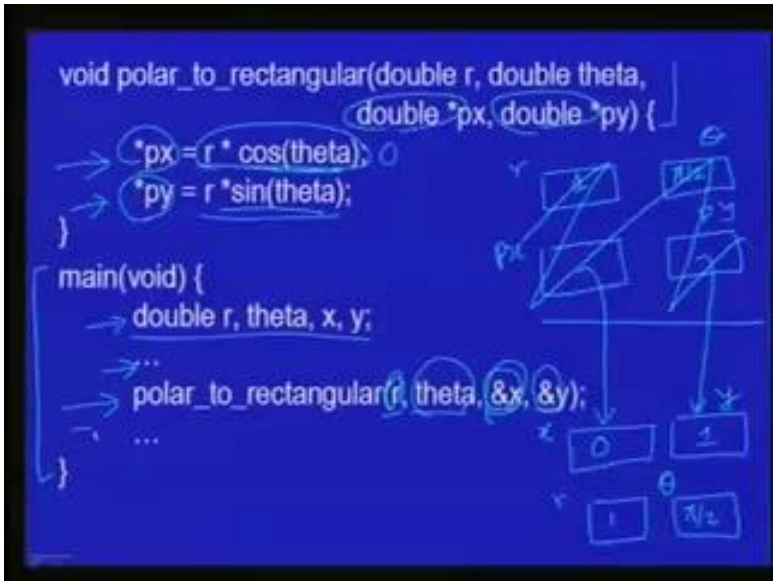
(Refer Slide Time: 19:50)



So, let us see what is the solution will look like the function polar to rectangular is similar to what we have written with some important differences. Now the type of the variables x and y which we are calling px and py to stand for pointers to x and pointers to y are double star instead of double. The parameters r and theta are remain of the type double. So, when we compute the value of x and y, r star cos theta is the x component and r star sin theta is y component instead of lining to x which would be illegal, because the variable x is not in scope, we assign it star px and to star py respectively. And when we call this function from main the variable declarations in main are the same as before.

So, we pass this parameter r and theta, we pass this argument r and theta as before. Now instead of passing this value of x and y, what we are passing are addresses of x and y. Note that ampersand x denotes the address of x or a pointer to x, and similarly ampersand y denotes the pointer to y, so that is what we are passing. So, let us try to understand in terms of pictures comprising of boxes and arrows what exactly is going on in this particular example. So, we have these when main starts executing the space for these variable gets created as we saw in the last lecture.

(Refer Slide Time: 21:23)



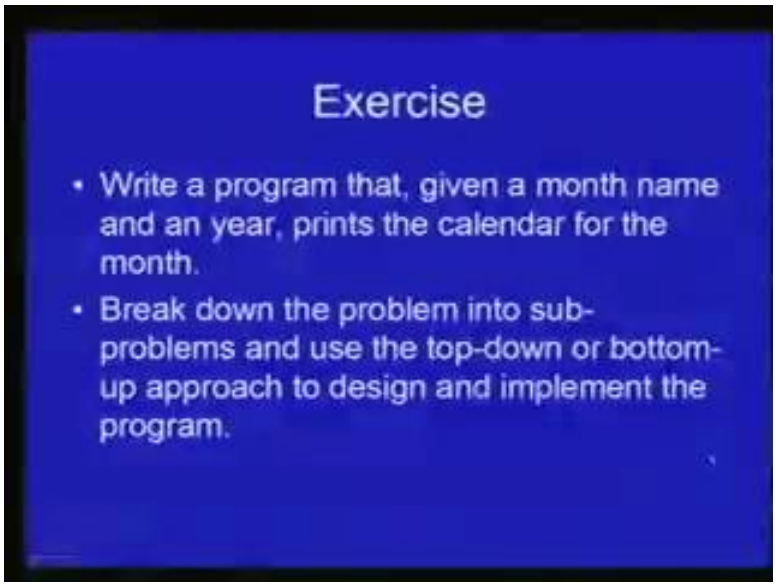
So, four variables get created r theta x and y and somewhere here r and theta are initialized. Let us assume that the value of r is one, just for example, and the value of theta is pi by 2. Now, when this function is called polar to rectangular r theta ampersand x ampersand y what exactly happens? As you know space will be created for the local

variables and the parameters of the function which is being called the function being called is polar to rectangular which has four parameters, but no local variables. So, the space for these four parameters will be created r theta, note that this r and this theta is nothing to do with r and theta of main and then we have px and py.

Now, these variables will be initialized by the values of these expressions, which are being passed as arguments. So, the parameter r gets the value of the expression r in main which was 1. So, this gets initialized to 1. Similarly theta gets initialized to the value of theta in main which is pi by 2. So, this gets initialized to pi by 2, what does px get initialized to px will get initialized to the value of this expression which is nothing but a pointer to x. So, px gets initialized to an arrow pointing towards the variable x of main. And similarly the variable py, the parameter py or the function polar to rectangular gets initialized to a pointer pointing towards the variable y of main.

So, this is what the picture looks like when the function polar to rectangular starts executing. Now this assignment takes place. So, right hand side is r into cos theta, r is one theta is pi by 2 and cos pi by 2 is zero. So, one into zero is zero. So, this entire thing evaluates to zero, and that gets assigned to star px. Now star px refers to the variable to which the px points which is x. So, the value zero is actually written here. And similarly next assignment computes the value of r sin theta, r sin theta will be one because sin pi by 2 is one and that value gets returned to the variable to which the parameter py points and that is the variable y, so the value one gets returned over here, and then the function polar to rectangular return. When the function polar to rectangular return, these variables get destroyed the control goes back to main, but when the control goes back to main note that the variables x and y have been modified to contain the rectangular equivalent of the polar coordinates r comma theta.

(Refer Slide Time: 24:41)



So, before we end this lecture today, here is an exercise for you to utilize the concepts that we have seen so far. So, should write a program that given a month name and a year, month name could be January, February and so on, and a year 1995, 2005 whatever should print the calendar for that month. And you should use the techniques that we have learnt so far that is divide the entire problem into sub problem that is identify the function possibly identify global variables. And then use either the top down or the bottom up approach and then start implementing the functions one by one, and finally, put them together as the main program to implement the entire solution.