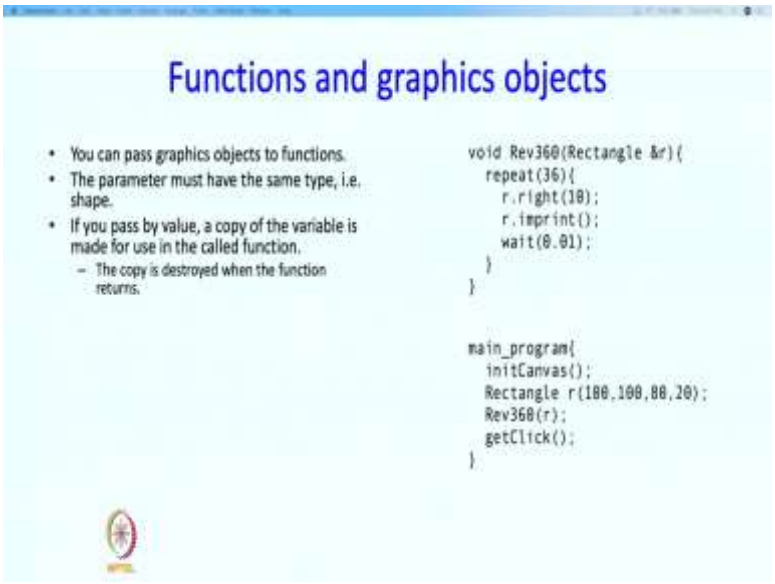
An Introduction to Programming through C++
Professor Abhiram G. Ranade
Department of computer science and Engineering,
Indian Information Technology Bombay
Lecture 10 (Part-5)
Functions Graphics Objects and Lectures conclusion

In the last segment we talked about pointers. In the next segment we will conclude this general discussion of functions, the basics of functions, but before that I want to say something about functions and graphics objects.

(Refer Slide Time: 0:38)



So here, I have a program in which I am using graphics and I am using a canvas on which I have rectangle created and then and calling that rectangle using a function Rev360 using this rectangle r, passing this rectangle r as an argument. So, I can do this and I want to explain how all this executes, ok. So, you can pass graphics objects to functions as happened over here. Now, a parameter must have the same type that is the same, it must be the same shape. So, indeed this r is a rectangle and, in fact, this r is also a rectangle. But not only is it a rectangle, but it is a reference parameter so that means this r will really be the same as this graphical object. So, whenever I referring to r in this body it will refer to the same graphical object that I had created in my main program. If I had not put this & over here than the usual rules apply, what are the usual rules? Whatever argument is this argument its value is copied into this.

So, at this point I want to tell you that when I talk about a graphics object. So say something like this r over here, this r is really doing some kind of double duty. So, nominally it is actually a variable also. So, it is a variable created in the activation frame of the main program, and it is actually the variable which contains all the information about that rectangle. I am not going to tell you what information it contains but just assume that it contains whatever information there is.

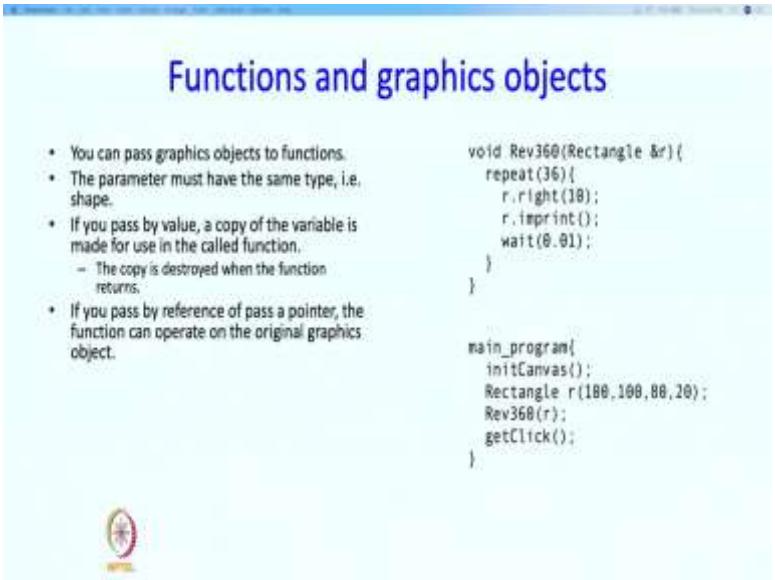
So, whatever information is needed to say draw it on the screen, so in some sense the drawing on the screen is kind of a by-product, so we are really creating a variable, when we write this we are creating something on the screen but we are also creating a variable. So that variable gets passed over here, but it is passed by reference, so really this r refers to the same graphical object on our screen. If, I passed it by value or if I did not have that and over here what would happen is, that a copy would get passed. So not only would the variable be copied but the way this graphic system has been designed, the graphical object will also be copied, when that copy is made initially the object will be at the same position as the object from which the copy has been made, but you would get a new copy. But that is not what is happening over here right now. we are getting, we are really passing the object by reference and therefore, this r is really the same as this rectangle over here.

(Refer Slide Time: 4:25)

So, what is this going to do, it is going to repeat but it going to rotate by 10 degrees, so then it is going to be imprint, so it will rotate and at every point it will imprint so what will happen

is, you will see a rectangle first, but it is going to be it will rotate and as it rotates it will imprint. So all these rotated positions imprinted on the screen, and so on.

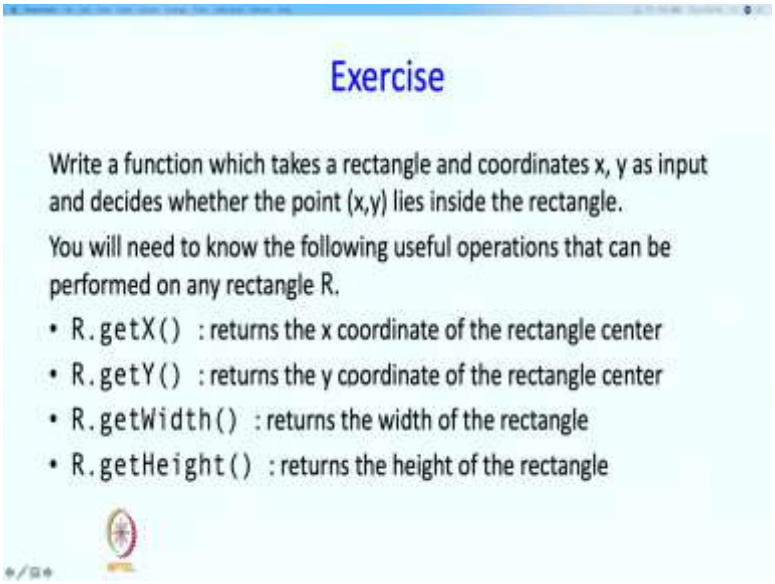
(Refer Slide Time: 4:47)



But at the end the rotation, the total rotation is 360 degrees because 36 times and every time we are rotating by 10 degrees, so at the end of original position will be attained. So it starts off in some position and at the end the same it will come back to the same position. But as it does that it will be imprinting, it will be imprinting on the screen. Now, the point is that the imprinting of the screen is going to survive. So even after, even after it returns the imprinting will stay there, alright. So, we said this if you pass by value a copy is of the variable is made and the copy of the graphic object is also made and I should say that as well, that if you made a copy of the variable and a copy of the graphics object, the variable as well as the graphics, the graphics object would get destroyed when you returned. Whereas in this case there is no destruction, so that r of the main program is going to stay around and that is the r that is going to get used up over here.

If you imprint that imprint happens on the screen and it will survive after the call finishes. And by the way I mean this copying business is allowed otherwise as well so I might as well state that here. So, for example here, I have created the rectangle by defining r but I create another rectangle s, and I say I want s to be exactly equal to r so I assign it. So, I can do that as well, so really when I make this copy is really sort of like an assignment statement. Anyway the point of this is that when I have graphics objects I can pass those graphics objects, and I can do things with them if I want.

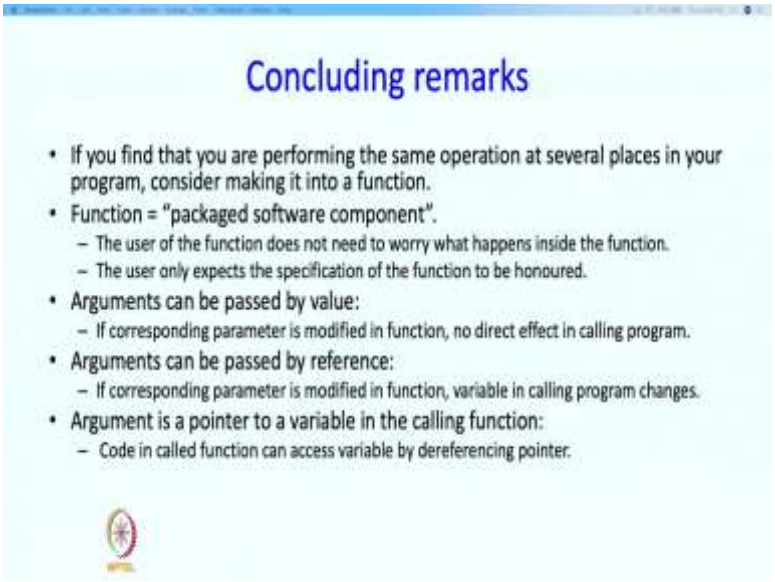
(Refer Slide Time: 6:43)



I would like you to think about a few exercises, so here is an exercise which might be quite interesting, might be quite useful if you are going to design some bigger programs. So you have to write a function which takes a rectangle and coordinates x, y as input and decides whether the points x, y lies inside the rectangle. So, this might be used to say that ok I am going to draw a rectangle on the screen and then I am going to do a get click, and I would like to know whether the user clicked inside that rectangle. So this function would come in handy. Now, you would have to pass that rectangle to our function and inside the function you want to get information about that rectangle and compare that with the x, y arguments that are also being passed. So, if you do R.getX() then you will get the x coordinate of the center of the rectangle, if you do R.getY() you get y coordinate of the center of the rectangle, if you do R.getWidth() you would get the width of the rectangle.

So, now you could see that by getting the width and the height and the x, y coordinates you can check whether the point x, y the x, y coordinates that have been given to you, for the point which lies inside the rectangle. So, this way you can see whether the user is clicking on a rectangle and if the user is clicking on a rectangle, may you could write a program which changes the colour of the rectangle, colour of that rectangle if the user clicks on it. So this could be a fun program, alright. So, as I said I am going to conclude in this segment but the conclusion is for the entire sequence of this lecture, so this basic lecture about functions.

(Refer Slide Time: 8:35)

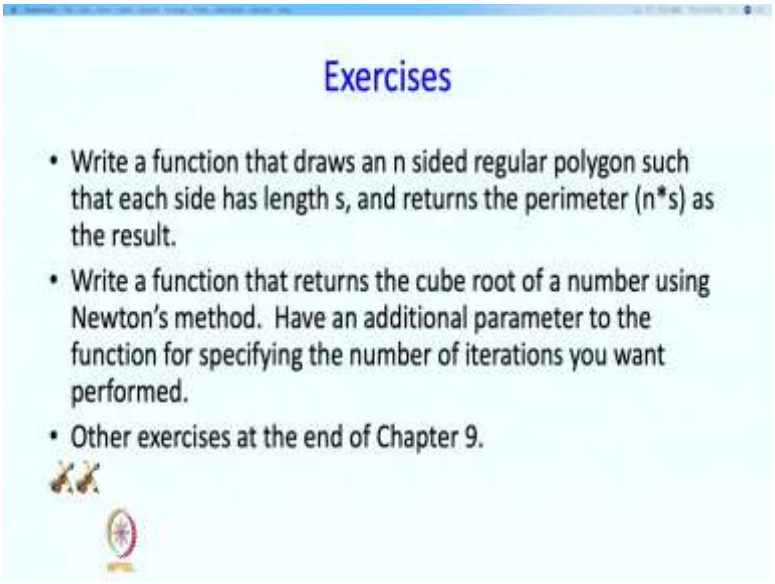


So, what did you say, we said that if you are find that you are performing the same operation at several places in your program, then consider putting it into a function, and do not duplicate code but make a call. You can think of a function as a contract that we talked about earlier but you can also think of it as a packaged software component, so just as you buy packaged hardware components. So anything that you buy practically is a packaged hardware components say you buy a television you do not know what is going on inside it, or if you buy a refrigerator you do not know what is going on inside it.

You are told, you are given some specifications and that exactly is how you should view functions, so its, its package packaged and its specification are told to you but you really do not need to look inside to see how that whole things actually works. You can just believe the specification you are expected to believe the specifications and if that does not work then you can of course sue the manufacturer or warranty period you can get things replaced, but that is what really all this is.

So, the function the packaged component idea is sort of like the contract idea that we talked about earlier. Then, we said that arguments can be passed by value, and here if the corresponding parameter is modified in the function no direct effect happens in the calling program. Arguments can be passed by reference, in which case if corresponding parameter is modified in the function, a variable, the variable in the calling program will change. Argument if the argument is a pointer to a variable in the calling function, then the code in the calling function can access the variable by dereferencing the pointer.

(Refer Slide Time: 10:32)



Now, there are lots something that you can do with functions, so for example, you can write a function that draws an n sided regular polygon such that each side has side length s, and returns the parameter of n times s as the result. Or you can write a function which that returns the cube root of a number using Newton's method. So, really all the algorithms, all the methods that we discussed in an earlier week, you can package it as functions, and then you can just make calls so you do not have to copy code any longer.

And there are other similar exercises given at the end of chapter 9 and I definitely suggest that you try them, so that concludes this basic lecture on functions and will stop here, thank you.