

An Introduction to Programming through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Lecture No 21 Part 5
Classes
Classes for graphics and input output

(Refer Slide Time: 00:18)

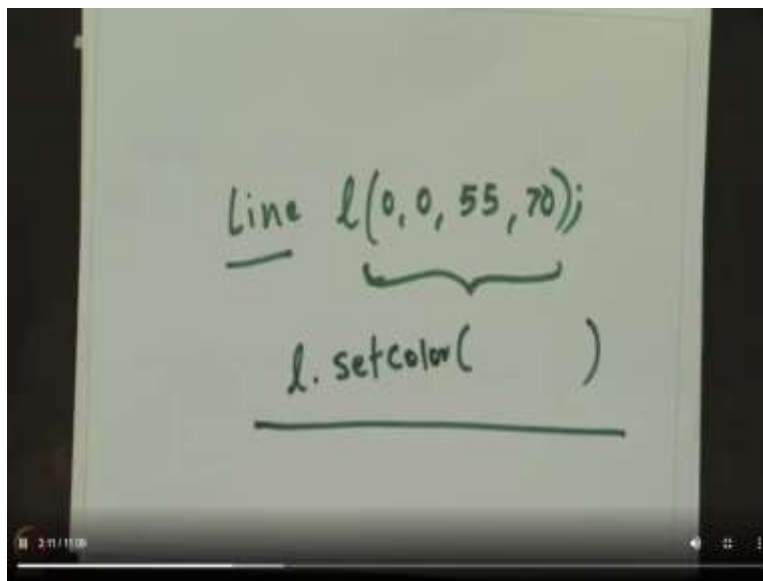



Welcome back, in the previous segment we talked about access control. In this segment, we are going to talk about some classes that you have really been using and well, maybe some which are slightly new but which you will want to use.

(Refer Slide Time: 00:35)

Graphics Classes

- You may have guessed that our graphics primitives such as Line, Rectangle, Circle are classes.
- When we create a graphics object, the constructor gets called.
 - Constructor initializes an area of memory to hold information about the object, but also draws the object on the graphics screen.
- There are data members which keep track of where the object is, its size, colour; however these members are private and you don't get to see them.



So, graphics classes you may have guessed that our graphics primitives such as Line, Rectangle, Circle are actually classes. So when we create lines we are really just creating, creating instances of that class or these are structs and we are creating structs.

And when we write something like `Line l(0, 0, 55, 70)`, this really is a constructor call. So in our graphics library, there is a class called Line or a struct called Line and there is a constructor which takes four arguments and this is really a constructor call and that returns a line object, or a line struct. So the constructor in this case is going to initialize an area of memory to hold

information about object, so which is what normal constructors do, they initialize the members. But, as we said earlier the constructors can contain arbitrary code as well. So in this case the constructor code is actually also going to draw the objects, object on the screen.

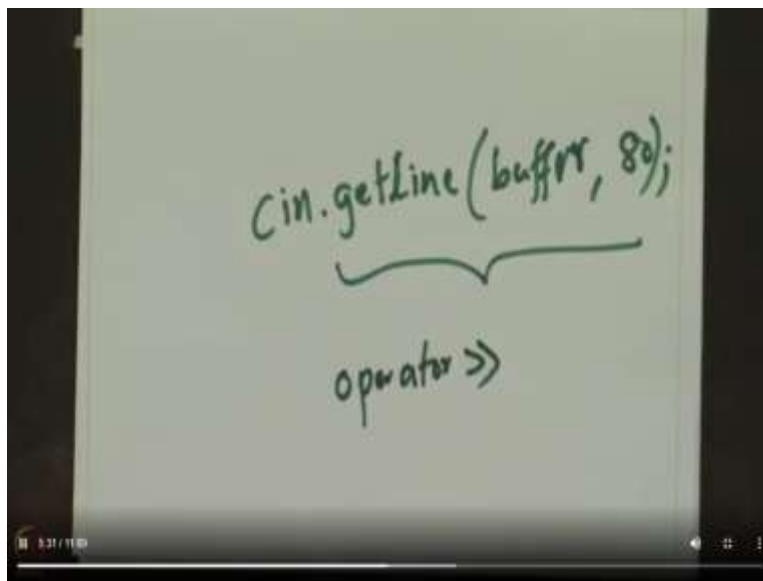

So, important point that the line creation commands are really constructor calls to a class called Line and similarly for circles, rectangles, text. Now in the graphics classes there are data members which keep track of things, like where is the graphics object? What is the size? So what is the size? What is the color and these members are private and therefore you do not get to see them.

How do you change the color? You have been given a call, you have been given a command of something but as you can see now this is just a member function. So you have been given member functions. You have been given that control panel which enables you to change colors. So you do not have to directly go in and change colors. Those are the colors. There is some information kept about the colors, but that is private information. And you do not want to know that because that information is kind of complicated. So instead of that it is better to have this have the commands which are convenient for you to see. And also graphics operations such as move and rotate are also member functions and how does your program get all of this? Well, the header file `simplecpp` fetches or includes all the class definitions when you include that file in your program.

(Refer Slide Time: 04:01)

Input output classes

- `cin`, `cout` : objects of class `istream`, `ostream` resp.
 - Need to include header file `<iostream>`
 - got included because you included `<simplecpp>`
- `<<`, `>>` : operators defined for the objects of these classes.
- `ifstream`: another class like `istream`.
- You create an object of class `ifstream` and associate it with a file on your computer.
- Now you can read from that file by invoking the `>>` operator!
- `ofstream`: a class like `ostream`, to be used for writing to files.
 - Must include header file `<fstream>` to use `ifstream` and `ofstream`.



Now like graphics classes, there also are input output classes, which you have been using, so `cin` `cout` are objects of class `istream` and `ostream` respectively and they are defined in the file `iostream`. If you remember we had a command, `cin.getline` of some buffer some buffer name care buffer name and maybe the length of the char buffer name. So now you can see by the syntax that this is a member function. So you have been using member functions already in graphics as well as in io.

Now you may say that oh iostream. I did not explicitly include but we have said we have discussed this earlier and we have told you that look that got included because you included simplecpp and << and >> are actually operators and they have been defined for the objects by using things like operator>>. So that is what it is, fstream is another class like istream and that is used for file io.

So you can create an object of class ifstream and associate it with a file on your computer. So once you do that, you can read from that file, pretty much like you read from cin and you can create an object of class ofstream and you can create an object of class ofstream which can be used for writing files. You know exactly a similar manner and end will see an example next but for this you have to include the header file fstream. So if you include that header file then you get to use ifstream and ofstream.

(Refer Slide Time: 06:21)

Example of file i/o

```
#include <fstream>
#include <simplecpp>
int main(){
    ifstream infile("f1.txt");
    ofstream outfile("f2.txt");
    repeat(10){
        int v;
        infile >> v;
        outfile << v<<endl;
    }
}
```

- Constructor call. object infile is created and associated with f1.txt, which must be present in the current directory.
- Constructor call. Object outfile is created and associated with f2.txt, which will get created in the current directory.
- Input and output can be performed using familiar >> and <<
- f1.txt must begin with 10 numbers. These will be read and written to file f2.txt.

So here is an example of file io. So this is our program. So as you can see it is including fstream. It is also including simple CPP because of all the usual things that we have been talking about. So namespaces, a namespace std and iostream and things like that. So what is new? So here is a definition, so we are defining an object infile and this is a constructor for it. Its type is ifstream. So we are defining an ifstream object.

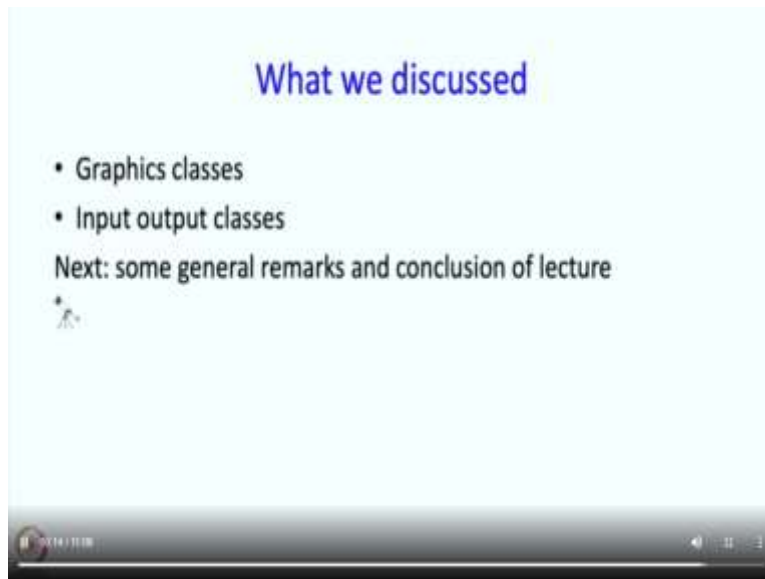
So what does this definition really do? So it says, it really says the following, it says that in my program the name in file is going to be there, so that is the ifstream, but while constructing it and associating it with the file f1.txt, which is going to be present in my directory. So that is what this line does. It is creating, it is a constructor call. So it is constructing a variable called infile and its type is ifstream. But the constructor is linking a file on your computer to this name inside your program. Similarly ofstream is a structure type and you are creating a variable or an object name outfile and type ofstream and the constructor is associating the file name f2.txt with this out file.

Now f2.txt is expected to be created when the program executes, it is an output file, a file. So it need not be there early on. And here is the main program, so inside the main program what are we doing? We are we have a variable v of type int and we are reading an integer from infile. So this operator works like before for cin so if I had said cin I would have extracted it would have extracted an integer from cin now it simply extracts an integer from infile.

So from this file in particular. And this would have sent the value of v to cout had had this been cout but now it is just going to send the value of v to outfile. That is it. So this is how you can read files and write files and there are some additional details that you might want to know, like maybe how do I append to a file and things like that. So that is discussed in the book. But we are not going to discuss it in this lecture. So anyway, so what does this program do?

So this program is going to read 10 numbers from infile and create a new file f2.txt and put those numbers inside that file. So this is of course trivial processing of files, but now you get the idea. So you should be able to write programs which take data from files and which put data into files not only take data from the keyboard and put data onto the screen.

(Refer Slide Time: 10:06)



So what have we discussed? We have discussed graphics classes, we have discussed input output classes and some of these things you have already been using but now, you know that the commands that you were using involved classes and constructors and member functions. Next, I am going to make some general remarks and then I will conclude this lecture, but before that, let us take a quick break.