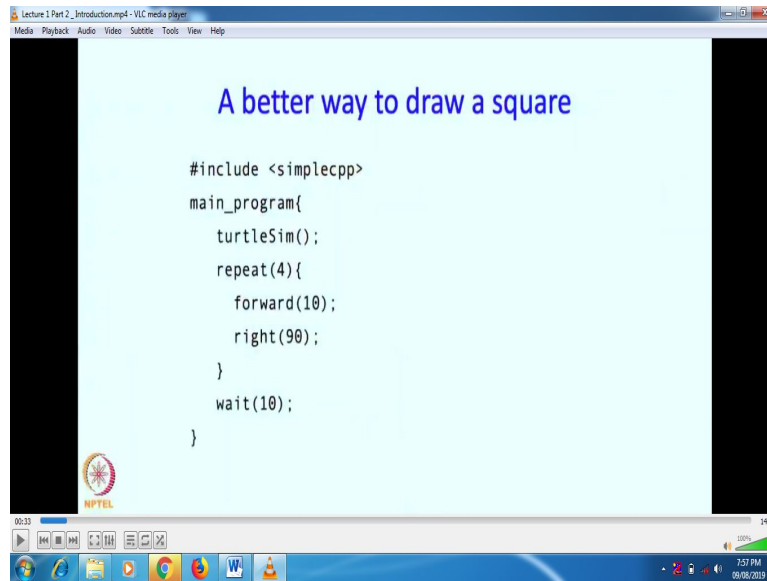


An Introduction to through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Lecture No. 1 Part - 2
Introduction

Repeat, input-output, and some basic commands

In the last segment, we discussed some general information about the course, how to install cpp and we wrote a programme for drawing a square.

(Refer Slide time 0:33)



The screenshot shows a VLC media player window titled "Lecture 1 Part 2_Introduction.mp4 - VLC media player". The main content is a slide with a light blue background and the title "A better way to draw a square" in blue text. Below the title is C++ code:

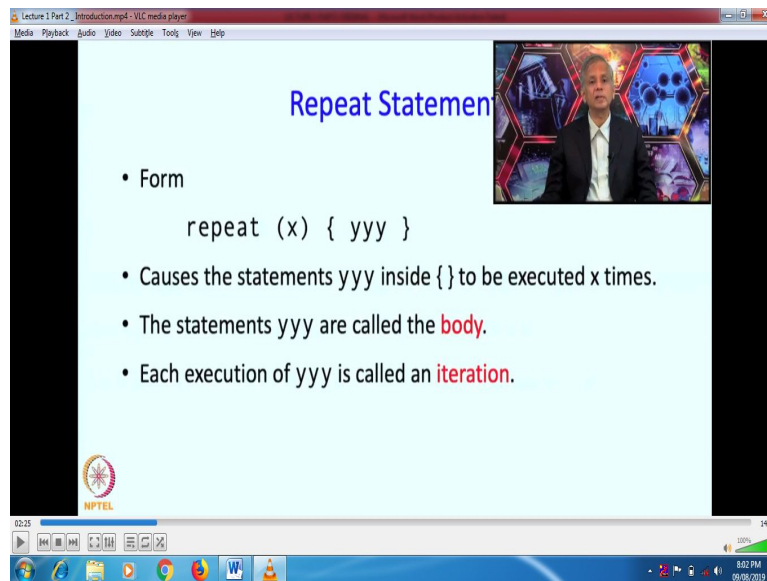
```
#include <simplecpp>
main_program{
    turtleSim();
    repeat(4){
        forward(10);
        right(90);
    }
    wait(10);
}
```

The slide also features the NPTEL logo in the bottom left corner. The VLC player interface includes a menu bar (Media, Playback, Audio, Video, Subtitle, Tools, View, Help), a progress bar, and a system tray at the bottom showing the time as 7:57 PM on 09/08/2019.

So now, I am going to show you a better way of drawing a square. So here, you can see that, we have the same first 3 lines but instead of typing in forward(10), right(90) several times, I have put it inside a 'repeat' statement, a so called repeat statement. So as you can see, after repeat there is four which tells you how many times to execute what is in the braces following the repeat.

So basically this statement is going to execute forward(10), right(90), 4 times and after that it is going to wait for 10 seconds. This will also draw a square but it is a nicer way to draw a square because I do not have type it so much. Imagine if I wanted to draw a 100-sided polygon, I would have had to type things 100 times. But with repeat statement, I can just say repeat 100 times.

(Refer Slide time 1:25)



The screenshot shows a video player window titled "Lecture 1 Part 2 - Introduction.mp4 - VLC media player". The slide content is as follows:

Repeat Statement

- Form

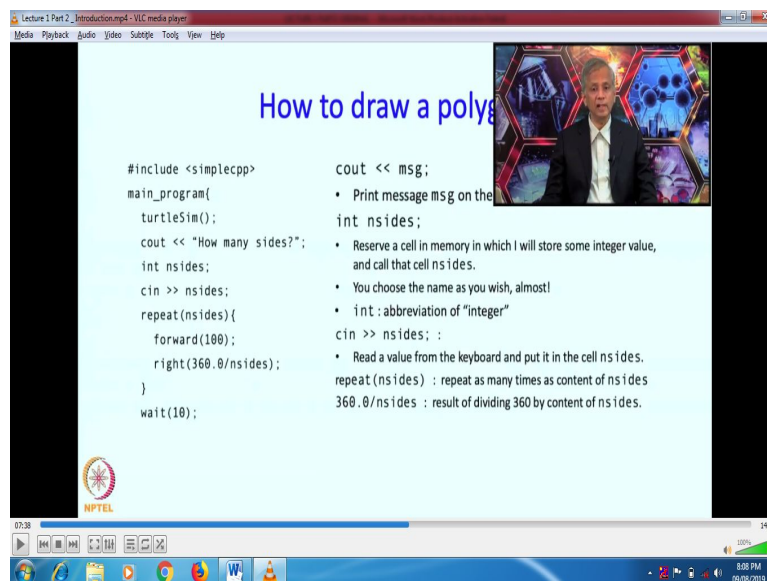
```
repeat (x) { yyy }
```
- Causes the statements yyy inside {} to be executed x times.
- The statements yyy are called the **body**.
- Each execution of yyy is called an **iteration**.

The slide also features an NPTEL logo in the bottom left corner and a small video inset of a man in a suit in the top right corner. The video player interface at the bottom shows a progress bar at 02:25 and a system tray with the date 09/08/2019.

Well, what does repeat statement do in general? And what does it look like? Its form in general is going to be this: ‘repeat’ followed by parenthesis inside which there is going to be a number, let us call it x and then there are these braces inside which there can be as many as statements as you like and collectively let us call these statements yyy. So what a repeat statement does? Well it causes these yyy statements inside the braces to be executed x times.

So we already saw an example, we had x equal to four and yyy was forward(100), right(90). So those two statements were executed 4 times. In general whatever is inside those braces is called the ‘body’ of the repeat statement and each single execution of yyy is called the iteration. So this repeat statement causes x iterations of the body yyy.

(Refer Slide Time 2:29)



The screenshot shows a video player window titled "Lecture 1 Part 2 - Introduction.mp4 - VLC media player". The slide content is as follows:

How to draw a polygon

```
#include <simplecpp>
main_program{
  turtleSim();
  cout << "How many sides?";
  int nsides;
  cin >> nsides;
  repeat(nsides){
    forward(100);
    right(360.0/nsides);
  }
  wait(10);
  cout << msg;
}
```

- Print message msg on the screen.
- Reserve a cell in memory in which I will store some integer value, and call that cell nsides.
- You choose the name as you wish, almost!
- int : abbreviation of "integer"
- cin >> nsides; : Read a value from the keyboard and put it in the cell nsides.
- repeat (nsides) : repeat as many times as content of nsides
- 360.0/nsides : result of dividing 360 by content of nsides.

The slide also features an NPTEL logo in the bottom left corner and a small video inset of a man in a suit in the top right corner. The video player interface at the bottom shows a progress bar at 07:38 and a system tray with the date 09/08/2019.

Now, we can use the repeat statement quite nicely to draw a polygon. So, here is the program and we will execute this later, but let us just see this. So it begins with include<simplecpp>

and `main_program` as before, then we start up turtle sim and now there is a different statement. So this statement is a 'cout' statement, its form is `cout << "message"`.

Now `cout` over here can be thought of as a command but you can also think of it as the screen. So this statement will print whatever you type after those less than less than on the screen. So in this case you typed "how many sides?", so this message how many sides will go on the screen.

After that, another statement that we have not seen: '`int nsides`'. `int nsides` is a rather complicated and interesting statement. So first of all, it tells the computer "please reserve a cell in memory for me in which I am going to store some integer value". How to store that value? I will tell you a little bit later, but what we are doing over here is we are reserving a cell in memory, not only are we reserving, but we are going to tell the computer that look from now on, if I use the term '`nsides`' you should treat it as referring to this cell. So essentially, I am giving `nsides` as the name for the cell that I just reserved.

Now you can choose the name as you wish, ok? We will see that in some later lecture and I just want to point out over here that '`int`' is an abbreviation of integer so you are telling the computer that I am going to store an integer and instead of writing down `i-n-t-e-g-e-r`, you just type `int` which is the abbreviation.

After that, there is another new statement '`cin >> nsides`'. Well what does this statement do? Well, it commands the computer to wait until the user types in a value from the keyboard. So the statement asks the computer to wait until a value is typed. So the user will see the message 'how many sides' and therefore, the user will type a value. So this value whatever it is, let us the user types a 10 that value 10 will come in into the computer and the `cin` statement will throw it into the cell '`nsides`' which you just reserved.

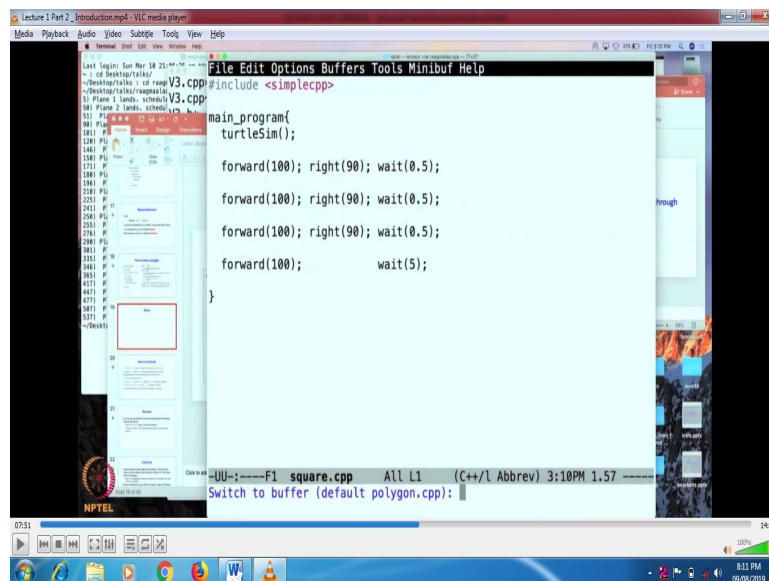
So think of '>>' than as arrows and think of `cin` as a representing the keyboard. So this statement is going to cause a value to flow from the keyboard to the cell in memory called '`nsides`'. Of course, this statement will cause the program to wait until you actually do the typing. If you do not type, then the program will just wait so you should type.

After that we have the 'repeat' statement, but now the repeat statement has gotten more interesting. We do not have a number inside the parenthesis, but we have the name of a cell. So what does this mean? So it says repeat as many times as whatever is the value that is contained in the cell `nsides`. So we just said that the user might have typed 10 when '`cin >> nsides`' statement executed. So if 10 was sitting in `nsides` then the loop, the body that is going to be following will be repeated 10 times.

So what is in the body? Well we are going to go forward 100 and now we are going to turn right but instead of turning right by 90 or some fixed value, we are going to turn right 360 divided by nsides. What is this 360 divided by nsides? Well it is a mathematical expression and you are instructing the computer to evaluate this mathematical expression. So, you are telling the computer divide 360 by whatever is contained in the variable nsides.

So, if we type 10 then nsides will contain 10, so as a result of this we will have 36 so the computer will turn 36 degrees. Notice that if there are 10 sides to a polygon then there are 10 exterior angles, if there are 10 exterior angles, each of which adds to 360 then each angle has to be 36. So this is exactly the angle that we need in order to draw a decagon. So this is what this repeat loop is going to do. After that the program will wait for 10 seconds and then the program is over.

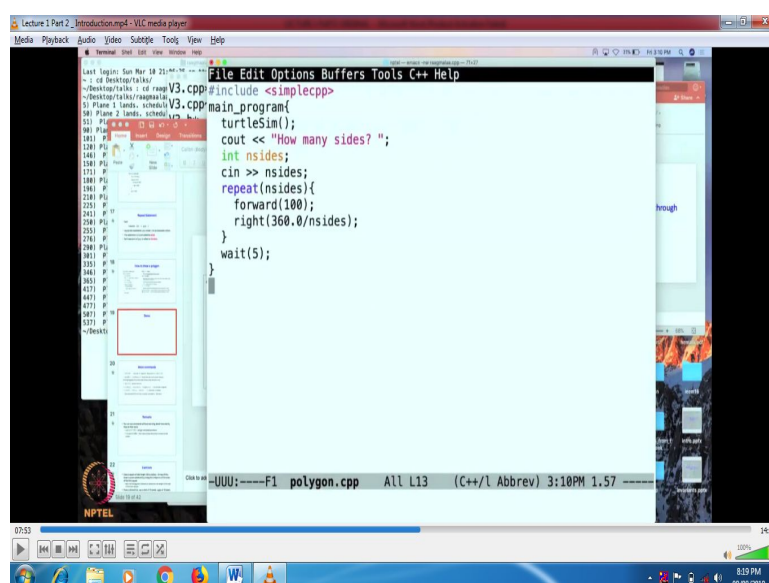
(Refer Slide Time 7:50)



```
File Edit Options Buffers Tools Minibuf Help
Last login: Sun Mar 18 21:14:11
~/Desktop/rahs/~/code/V3.cpp#include <simplecpp>
main_program{
  turtleSim();

  forward(100); right(90); wait(0.5);
  forward(100); right(90); wait(0.5);
  forward(100); right(90); wait(0.5);
  forward(100);      wait(5);
}
```

Switch to buffer (default polygon.cpp):

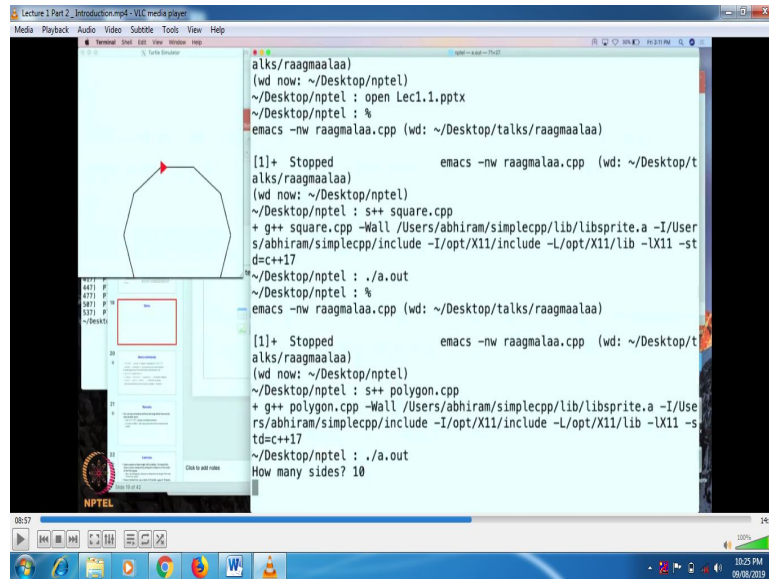


```
File Edit Options Buffers Tools C++ Help
Last login: Sun Mar 18 21:14:11
~/Desktop/rahs/~/code/V3.cpp#include <simplecpp>
main_program{
  turtleSim();
  cout << "How many sides? ";
  int nsides;
  cin >> nsides;
  repeat(nsides){
    forward(100);
    right(360.0/nsides);
  }
  wait(5);
}
```

All right, so let us again see a demonstration of this. So this is the same editor and this is the program that we just saw. So I am going to go to the new program – ‘polygon.cpp’, which is

what we just wrote. So you can see it is reserving a cell in memory called nsides and these cells in which we store values are often called variables. And then it is turning right by 360 by nsides and the repeat loop goes nsides times. I put in a wait 5 over here whereas earlier I had said wait 10 does not really matter.

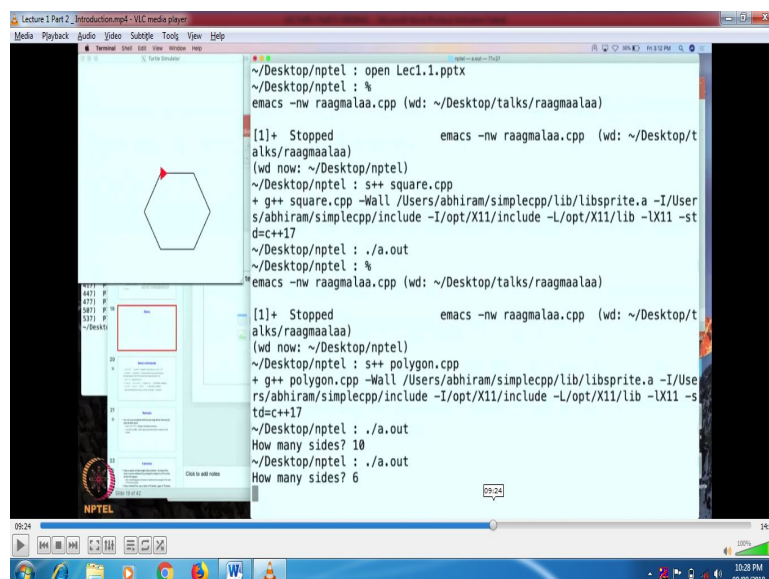
(Refer Slide Time 8:21)



So, let us compile that programme. So this is polygon, and let us execute. So we have our turtle, so it has asked us a question, “How many sides?” I am going to respond 10. Let me just move this so that I can see both. So I am going to type 10, but the statement does not work just by typing 10, I have to hit a return as well. So now, it ended up drawing a polygon.

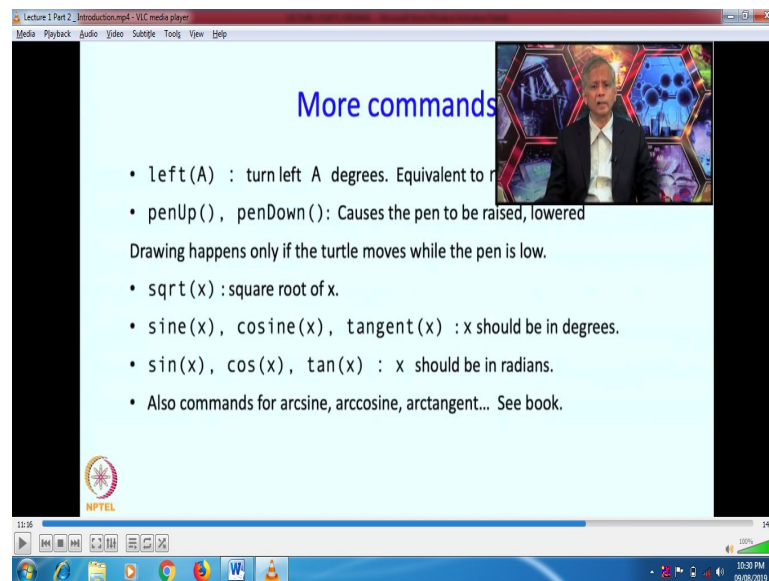
Well there is some problem over here, that polygon was little too big and did not fit in that window we have. Well that is not a big deal. Even if the polygon part of the polygon went outside, the turtle knows how to come back.

(Refer Slide Time 9:18)



But let us just execute it one more time, so that you can see a full polygon. So this time, when it says “how many sides?”, let me type 6. So now, it draws a hexagon and you will see that it has turned exactly the right amount. How did it know how much to turn? Well, it turned 360 by nsides and so in that case it turned 360 by 6 which is 60 degrees. Ok, so we have seen one more program to draw a polygon and I just now want to point out that there are many more commands available at our disposal. So we do not have to always turn right we can turn left.

(Refer Slide Time 9:51)



And if you want to turn left through A degrees, we can say turn left A degrees. Well actually we do not really need that left command, because if we say ‘right(-A)’, then that is really equivalent, but just for convenience that left command is also provided.

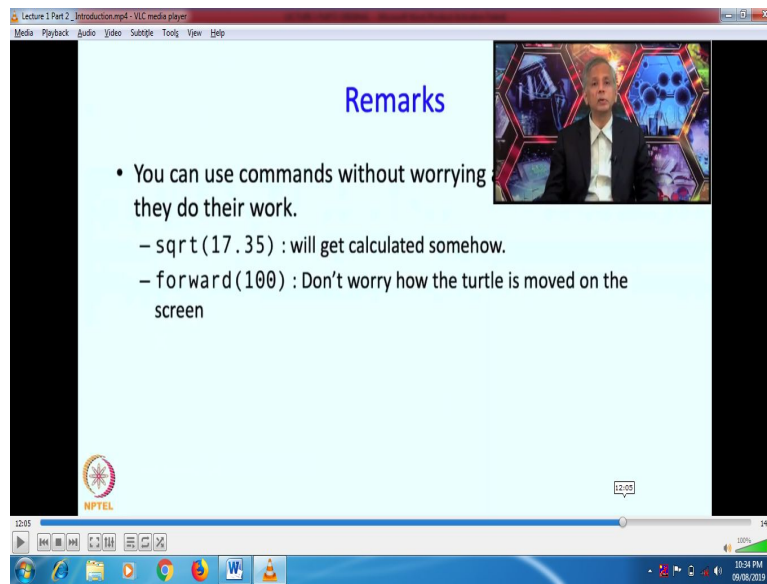
Two additional commands are there - ‘penUp()’ and ‘penDown()’, and these respectively cause the pen to be raised and lowered. What does that mean? Well the drawing is going to happen only if the turtle moves while the pen is low. If you raise the pen, if you issue the ‘penUp()’ command, and then move, only the turtle will move, no drawing will be done, so after that you need to put down the pen again by giving the ‘penDown()’ command, and only then will any drawing happen (from that point onwards)

You can do mathematical calculations, so for example, you can use the commands ‘sqrt(x)’. So this is going to cause the square root of x to be computed, and instead of this square root of x, essentially that actual square root will appear and so, then you can use it directly in your calculations.

Similarly you can use sine(x), cosine(x), tangent(x) and so on. So here ‘x’ should be in degrees. In scientific computation it is more customary to use radians, so actually, the

abbreviated forms are allowed. There also are commands for arcsine, arccosine and lots of useful mathematical functions, so for that, please see the book.

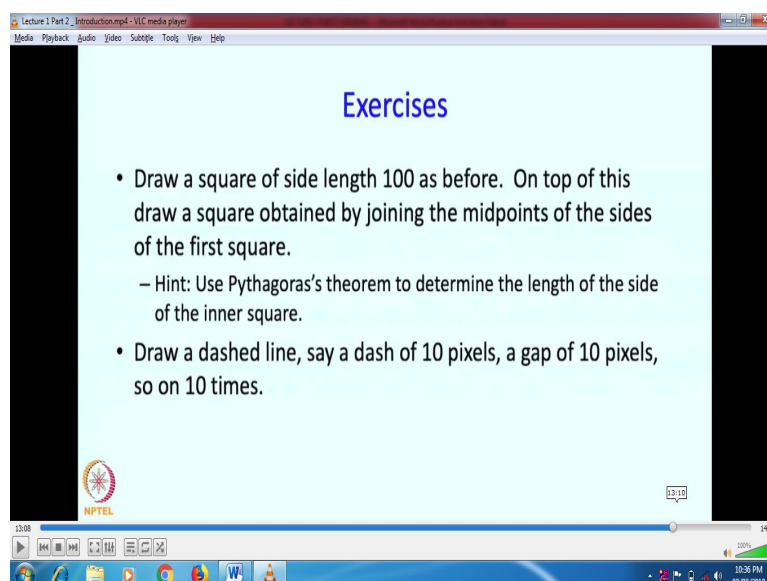
(Refer Slide Time 11:24)



Okay, so some remarks. Now, you can use commands without worrying about exactly how they do the work. So if I write `sqrt(17.35)`, or indeed whatever number, it will get calculated somehow, you do not have to worry about it. The computer knows already, how to calculate square roots.

Similarly, when I say `forward(100)`, the computer already knows how to move the triangle forward 100, so you do not have to worry about exactly in which direction it is, and so on. So that is the point of what happens when when we talk about a command, it promises that it will do something and you do not have to worry about how that something is actually done, it will just happen.

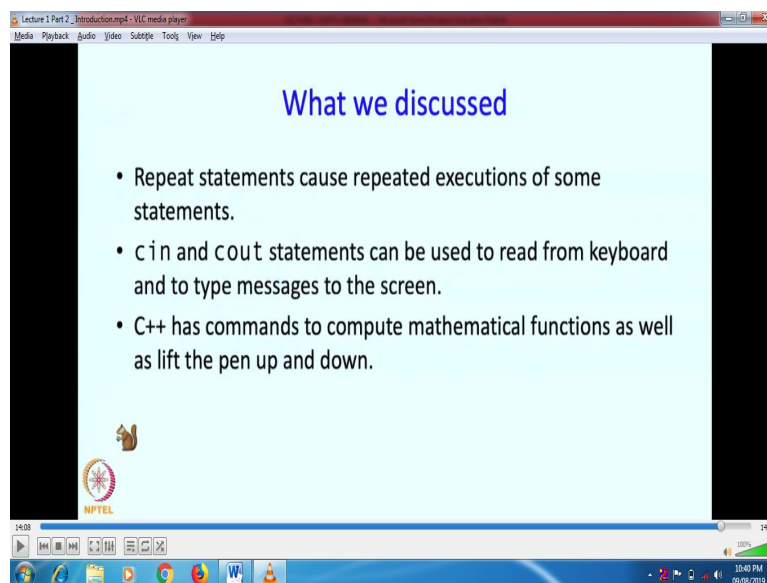
(Refer Slide Time 12:12)



So here is a small exercise based on what we have seen so far. So draw a square of side length 100 as before, but, on top of this, you are to draw a square obtained by joining the midpoints of the sides of the foursquare. So now, you will have to calculate what the side lengths of this new square are. So now, you can use Pythagoras' theorem to determining the length of the sides of the inner squares and then you will typically need to do a square root calculation, but you know how to do square roots and so you can use that to decide how much the turtle is supposed to move.

Here is another exercise, instead of drawing a straight line, draw a dash line say for example a dash of 10 pixels, then gap of 10 pixels and so on 10 times. So what this will require you to do, is you will have to raise and lower the pen 10 times, but, DO NOT, and I emphasize, DO NOT write these commands 10 times, put them inside a repeat statement. So that is that is really a very important part of saying that I know programming. One of the major things about I know programming is that I know how to use repeat statements.

(Refer Slide Time 13:55)



So what have we discussed? We discussed the 'repeat' statement, which is a really important statement which causes repeated executions of some statements which are in its body. 'cin' and 'cout' statements also we discussed, and these can be used to read from the keyboard and to type messages to the screen. C++ has commands to compute mathematical functions as well as lift the pen up and down. So, we will stop here for a short break.