

**Programming in C++**  
**Prof. Partha Pratim Das**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 20**  
**Classes and Objects (Contd.)**

Welcome to part-2 of module 11, of Programming in C++. In the earlier part of this module, we have seen the basic concepts of classes and instances of classes as objects. We have understood, what are data members and methods. We have discussed three examples particularly of complex numbers; of rectangle objects with points and of a stack. We have also understood this pointer for identifying an object.

In the remaining part we would briefly discuss what is known as the State of an Object.

(Refer Slide Time: 00:58)

**State of an Object: Complex**

- The state of an object is determined by the combined value of all its data members. Consider class Complex:

```
class Complex { public:  
    double re_, im_; // ordered tuple of data members decide the state at any time  
  
    double get_re { return re_; }  
    void set_re(double re) { re_ = re; }  
    double get_im { return im_; }  
    void set_im(double im) { im_ = im; }  
};
```

Complex c1 = {4.2, 5.3};  
// STATE 1 of c1 = {4.2, 5.3} // Denotes a tuple / sequence

- A method may change the state:

```
Complex c = {4.2, 5.3};  
// STATE 1 of c = {4.2, 5.3}  
  
c.set_re(6.4);  
// STATE 2 of c = {6.4, 5.3}  
  
c.get_re();  
// STATE 2 of c = {6.4, 5.3} // No change of state  
  
c.set_im(7.8);
```

The diagram shows a box divided into two sections. The top section contains the value 4.2, and the bottom section contains the value 5.3. Handwritten labels 're' and 'im' are next to the top and bottom sections respectively, and 'Double' is written next to each value.

This is a notion which is derived from the way the object oriented paradigm applies in the contexts of C++. We say that the state of an object is determined by the combined value of all its data members.

In simple terms, let us say that if we are going back to the complex example. So, this data part is same of course, we have some additional methods just for the purpose of demonstration. So, these data members what do they say? They say that if I have a complex number, if I can just draw it out then a complex number is, it has a 're' component then and an 'im' component. So, this is a complex number. So, if I say c 1 is defined initialized to be 4.2, 5.3 then I can say this is c 1 and this is 4.2 and this is 5.3.

So, we say the state, notion of state is double value; this also has a double value. So, I know that I can have any double value as the value of re. Similarly, I can have any double value as a value of im. So, every possible pair of double value that I can put to re and to im will denote a different complex number. So, if I change in c 1, if I change this value or if I change this value or if I change both the values. So, we will say that c 1 has acquired a different state.

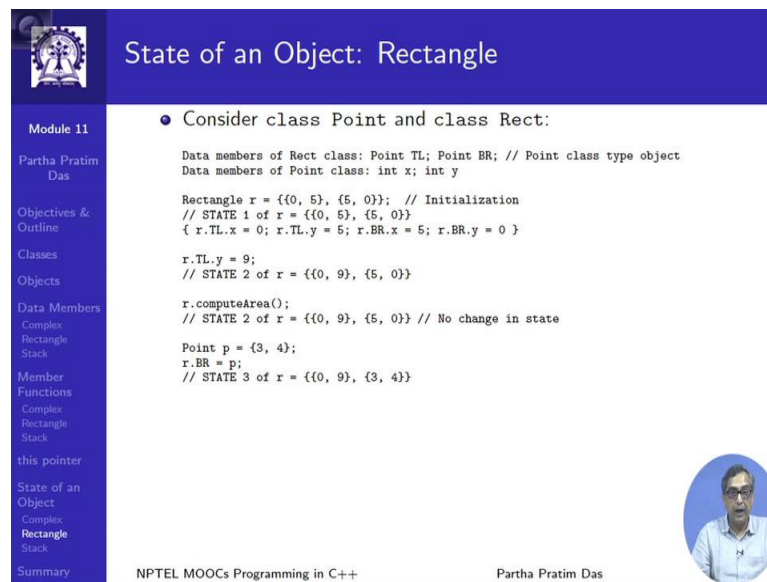
As long as the data members of an object maintain a certain combination of values, we say it is in one state and as soon as any one of the data members change in its value, we say it is in a different state. So, it is looked at this way that finally, in programming, it is about deciding on what state the object is in and with the use of methods, what is the next state the object could get into. We will look at all this dynamics subsequently more, but this is just to show you that, what is the notion of state in depth?

To show that we just make a use of four methods here, just look at them closely, get re; basically reads returns the component re. So, if c 1 is 4.2, 5.3 and if I invoke c 1 dot get re I will obviously, get 4.2. Similarly, if I do get im, it will return me 5.3 and the other two these are set. So, basically if I pass a value as re to the set re method, then it will set that value into the re underscore component of the object on which I have called set re. So, these are commonly called get set method, we will look at them more. So, with that let us see, what are the different states that I can be into? So, this is the initial one, it is initialized with 4.2, 5.3. So, state 1 of c is 4.2, 5.3, since there are two data members the state is defined in terms of doublet, in terms of a pair of numbers here.

Then, if I do c dot set re 6.4, then the 4.2 changes to 6.4. So, I have a new state two which is 6.4, 5.3 here, that is state 2. Now, in this let us suppose I invoke c dot get re c

dot get re, which will read the re component of the c object which is basically, now 6.4. So, it will return 6.4, but you can note that, there is no change in the re or im components of the object. Therefore, you will conclude that there is no change in the state, so it continuous to be in state 2. But if I do set im 7.8, then naturally my state will change because now the object will become 6.4, 7.8. So, in this way as different operations are performed on an object, it goes through different states and we always say that the data members in a way remember the state of an object.

(Refer Slide Time: 06:03)



The slide is titled "State of an Object: Rectangle" and is part of Module 11. It contains C++ code demonstrating the state of a Rectangle object through three different states. The code defines a Point class and a Rectangle class. The Rectangle class has two Point objects as data members: TL and BR. The code shows the initialization of a Rectangle object r, followed by changing the TL point to (0, 9) and then the BR point to (3, 4). The computeArea() method is called, and the state of the object is noted as "No change in state" after the area computation.

```
• Consider class Point and class Rect:

Data members of Rect class: Point TL; Point BR; // Point class type object
Data members of Point class: int x; int y

Rectangle r = {{0, 5}, {5, 0}}; // Initialization
// STATE 1 of r = {{0, 5}, {5, 0}}
{ r.TL.x = 0; r.TL.y = 5; r.BR.x = 5; r.BR.y = 0 }

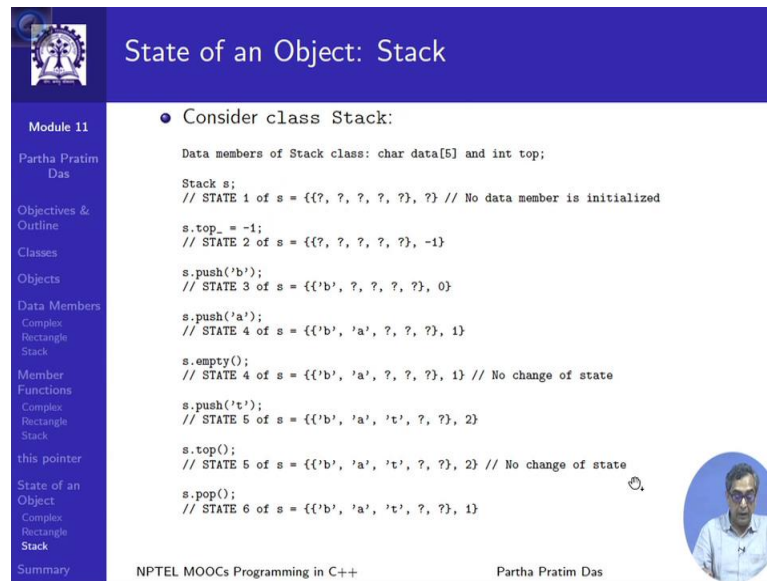
r.TL.y = 9;
// STATE 2 of r = {{0, 9}, {5, 0}}

r.computeArea();
// STATE 2 of r = {{0, 9}, {5, 0}} // No change in state

Point p = {3, 4};
r.BR = p;
// STATE 3 of r = {{0, 9}, {3, 4}}
```

These are more examples for you to practice. This is an example showing a rectangle, it is corner points and as we changed different coordinates of the corner points or we compute ratio, we show how the state of object is changing.

(Refer Slide Time: 06:22)



The slide is titled "State of an Object: Stack" and is part of Module 11, "Programming in C++", by Partha Pratim Das. It illustrates the state of a Stack object through a series of operations. The code defines a Stack class with a character array 'data' of size 5 and an integer 'top'. The state of the object is shown as a tuple of the array and the top value. The operations and their effects are as follows:

- STATE 1:** Initial state: `{ '?', '?', '?', '?', '?' }`, `top = ?`. Comment: "No data member is initialized".
- STATE 2:** After `s.top_ = -1;`: `{ '?', '?', '?', '?', '?' }`, `top = -1`.
- STATE 3:** After `s.push('b');`: `{ 'b', '?', '?', '?', '?' }`, `top = 0`.
- STATE 4:** After `s.push('a');`: `{ 'b', 'a', '?', '?', '?' }`, `top = 1`.
- STATE 4 (continued):** After `s.empty();`: `{ 'b', 'a', '?', '?', '?' }`, `top = 1`. Comment: "No change of state".
- STATE 5:** After `s.push('t');`: `{ 'b', 'a', 't', '?', '?' }`, `top = 2`.
- STATE 5 (continued):** After `s.top();`: `{ 'b', 'a', 't', '?', '?' }`, `top = 2`. Comment: "No change of state".
- STATE 6:** After `s.pop();`: `{ 'b', 'a', 't', '?', '?' }`, `top = 1`.

The slide also includes a navigation menu on the left and a small portrait of the instructor in the bottom right corner.

There is another example on stack. So, in stack what did we have? We have one data array and the indicated top. So, the state will comprise the whole array. So, if the data array has size 5, then it has a 5 quintuple denoting all the characters that, this data array has and then another component top which is the value of top. So, these together will give me the state of s and these are all question marks because at the beginning nothing has been initialized. So, I do not know what state the object is in.

But, then as I perform initialization to the top, this becomes minus 1. So, I get some state, but I still do not know, what is the state of the array I push b. So, the first character becomes b this gets incremented to 0, my state changes I push a, 'b' 'a' this changes. When I check for empty it is not empty and it returns me false and it does not change the array of the top. So, there is no change of state and so on.

So, if you just proceed to follow this you will see that with the operations of push and pop, we will actually be changing the state, whereas with top and empty we will not be changing the state and at any point the stack can be described in terms of the state of its array and the state of top marker this is to. So, you would have frequently heard about objects having states. So, this is the basic meaning of the state as you go forward, we will discuss more about the behavior of on object.

(Refer Slide Time: 08:13)

### Module Summary

- We have covered the following:

Class	<pre>class Complex { public:     double re_, im_      double norm() { // Norm of Complex Number         return sqrt(re_ * re_ + im_ * im_);     } };</pre>
Attributes	<pre>Complex::re_, Complex::re_im_</pre>
Method	<pre>double Complex::norm();</pre>
Object	<pre>Complex c = {2.6, 3.9};</pre>
Access	<pre>c.re_ = 4.6; cout &lt;&lt; c.im_;</pre>
this Pointer	<pre>double Complex::norm() { cout &lt;&lt; this; return ... }</pre>

NPTEL MOOCs Programming in C++ Partha Pratim Das

So, with this we will close on the module 11. In module 11, we have covered the following we have understood the basic concept of a class with data members and methods. We have seen that the attributes or data members can be named in the name space of complex. So, re, the name of re underscore actually is complex colon colon re and so on.

The method is also named similarly in the name space of complex. So, a method norm has a name complex colon colon norm. The objects are instantiations of the classes and they can be initialized when they are instantiated, axis is done in terms of using the dot operator and there is a special this pointer which identifies every object by its own address, which can be used in different methods.