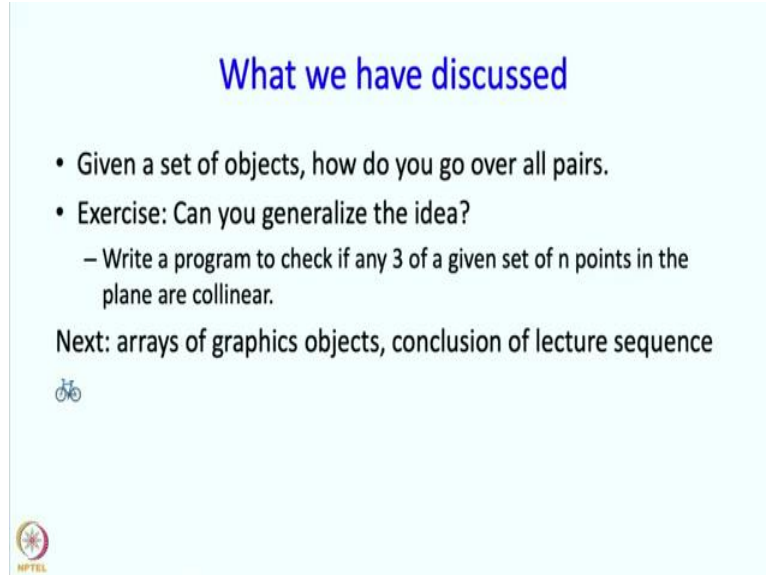


An Introduction to Programming through C++
Professor. Abhiram G. Ranade
Department of Computer Science and Engineering,
Indian Institute of Technology Bombay.

Lecture 15

Array Part-1 Arrays of Graphical Objects and Conclusion



(Refer Slide Time: 0:20)



What we have discussed

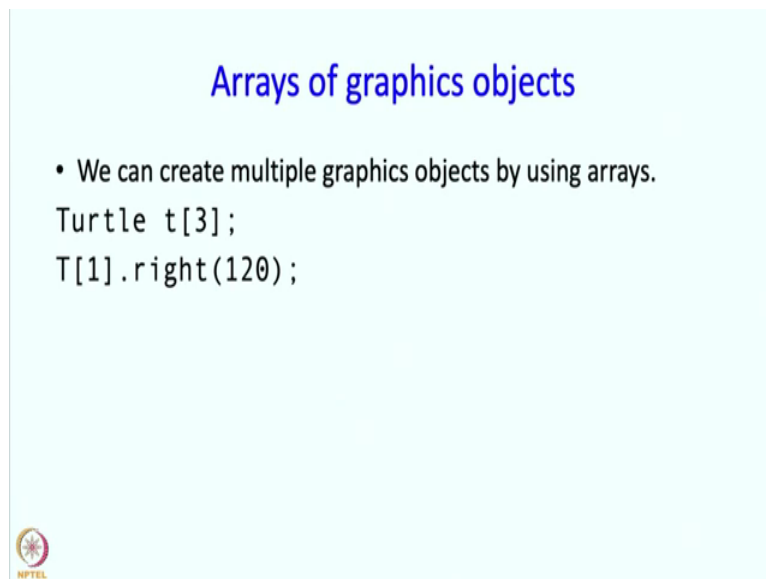
- Given a set of objects, how do you go over all pairs.
- Exercise: Can you generalize the idea?
 - Write a program to check if any 3 of a given set of n points in the plane are collinear.

Next: arrays of graphics objects, conclusion of lecture sequence



Welcome back, in the last segment we considered the problem of deciding whether a set of disks is intersecting or not. In this segment we are going to look at arrays of graphics objects, whether that is even possible, and we are also going to conclude this entire lecture sequence. So, can we have arrays of graphics objects?


(Refer Slide Time: 0:46)



Arrays of graphics objects

- We can create multiple graphics objects by using arrays.

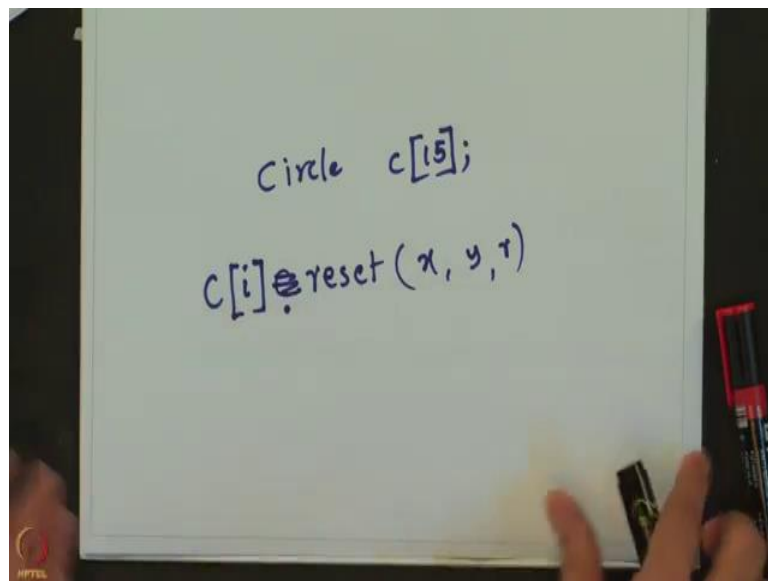
```
Turtle t[3];  
T[1].right(120);
```



Indeed, we can. So, if I create an array, we will end up creating multiple graphics objects. So, for example, I can write Turtle t, and that gets me an array of 3 Turtles. So, if I write such a statement, this statement has to appear after I write initCanvas(). Because when I create the array of turtles, the turtles will be created on the canvas as well.

So, such a statement has to appear after I have executed initCanvas(). And furthermore, if I create such an array, I will get 3 Turtles and they will be created at the same position, as what we get when we create a single turtle. Which is, all of them will be created sitting at the centre of the screen. And furthermore, they will all be facing right.

(Refer Slide Time: 2:14)



So, in a similar manner, I can also say create circles. I can write say, Circle c[15]. Now there is a slight difference between Turtles and Circles. So, when I create a circle I am expecting to give it centre and radius. So, when I create an array, there is no room for me here in this syntax to give the centre and the radius. So, what I need to do when am creating an array of circles, or indeed an array of lines or rectangles, is that is subsequently I have to execute c.reset, with x, y and r for an individual circle or, for a circle from a circle from an array I have to issue, c[i].reset(x, y, r). And here I can specify the x and y co-ordinates of the centre of the ith circle, and also the radius of the ith circle.

Reset command is discussed in chapter 5 of the book. So, this for circles, will indeed create and array of circles, but they will not have radii, they will not have centres, they will be dummy circles until you do this reset.

(Refer Slide Time: 3:47)

Arrays of graphics objects

- We can create multiple graphics objects by using arrays.

```
Turtle t[3];  
T[1].right(120);
```



But coming back to Turtles, if I issue this you will indeed get turtles on the screen, but it will be sitting in the same position, which is the centre of the screen and all pointing to the right. Now, I can talk to one Turtle and I can write, T[1].right, which is what you might expect. So, this will turn the first turtle, remember there is a 0 turtle as well, which will turn the first turtle right by 120 degrees.

(Refer Slide Time: 4:19)

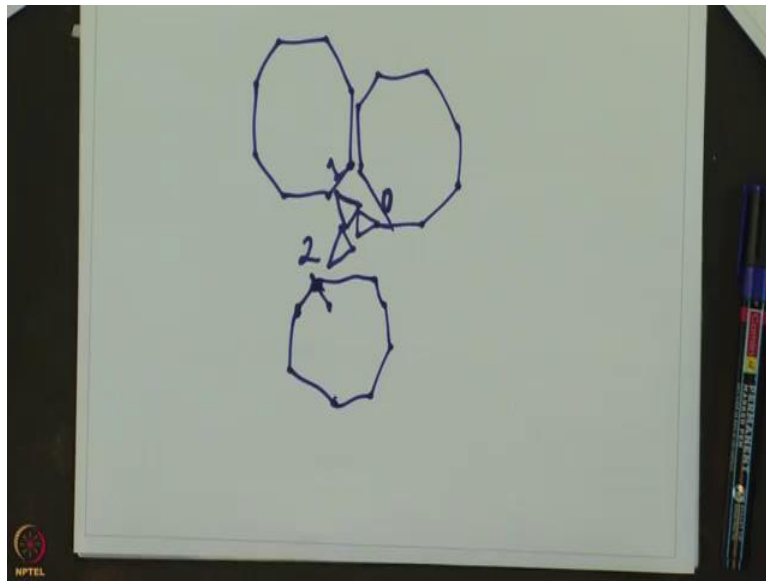
A program

```
int main(){  
    initCanvas();  
    Turtle t[3];  
  
    t[1].left(120);  
    t[2].left(240);  
  
    repeat(8){  
        for(int i=0; i<3; i++) t[i].forward(100);  
        for(int i=0; i<3; i++) t[i].left(360.0/8);  
    }  
    wait(5);  
}
```



So, here is a programme, let us see what it does. So, it is going to first create the canvas, then it is going to have turtles created. Then it will turn T1 left by 120, T2 left by 240. So, what is this going to do.

(Refer Slide Time: 4:43)



A program

```
int main(){
    initCanvas();
    Turtle t[3];

    t[1].left(120);
    t[2].left(240);

    repeat(8){
        for(int i=0; i<3; i++) t[i].forward(100);
        for(int i=0; i<3; i++) t[i].left(360.0/8);
    }
    wait(5);
}
```

So, I have 3 turtles one on top of the other. `t[1]` is going to turn left by 120, so one of the turtle is going to start facing in this direction. So, this is turtle 1. The other turtle is going to turn by 240, so it is going to face in this direction, this is going to be turtle 2. And turtle 0 is going to be facing in the same direction. We have not changed the orientation of turtle 0.

Next I have a repeat statement, it is a little bit complicated, because inside it there is a for loop. So, inside, am going to do something for every turtle. And for every turtle am going to do same thing, forward 100 and left 360 upon 8. Do you remember what this does? If this had been just forward 100 and left 360 upon 8, this would have been drawing an octagon, with side length 100.

So, effectively, each turtle is going to draw an octagon. But this turtle is pointing in this direction. So, it is going to draw an octagon which looks something like this. This turtle is going to draw an octagon, which looks something like this. And this turtle is going to draw an octagon, which looks something like this, so starting over here, it is going to go forward then left. So, that is what it is going to do and then it is going to wait for you, to see what has happened, and that is the end of it. So, let us see this in action.

(Refer Slide Time: 6:35)

```

File Edit Options Buffers Tools C++ Help
#include <simplecpp>

int main(){
    initCanvas();
    Turtle t[3];

    t[1].left(120);
    t[2].left(240);

    repeat(8){
        for(int i=0; i<3; i++) t[i].forward(100);
        for(int i=0; i<3; i++) t[i].left(360.0/8);
    }
    wait(5);
}

-UU-;-----F1 3poly.cpp All L13 (C++/L Abbrev) 4:53PM 1.44

```

```

+ g++ generalRollNos.cpp -Wall /Users/abhiram/simplecpp/lib/libspite.a -I/Users
/abhiram/simplecpp/include -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/npTEL/week7 : ./a.out
-1
~/Desktop/npTEL/week7 : more generalRollNos.dat
190050021 65
190050010 42
197050015 78
190060037 91
180030001 88
190370051 91
180020041 66
190010045 44
190030101 83
190120077 91
~/Desktop/npTEL/week7 : ./a.out
190370051
91
190380051
Roll number not found.
-1
~/Desktop/npTEL/week7 : %
emacs -nw highest.cpp

[1]+ Stopped emacs -nw highest.cpp
~/Desktop/npTEL/week7 : s++ 3poly.cpp

```

```

week7 : more generalRollNos.dat
week7 : ./a.out
91
190380051
Roll number not found.
-1
~/Desktop/npTEL/week7 : %
emacs -nw highest.cpp

[1]+ Stopped                  emacs -nw highest.cpp
~/Desktop/npTEL/week7 : s++ 3poly.cpp
+ g++ 3poly.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/
simplecpp/include -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/npTEL/week7 : ./a.out

```

```

~/Desktop/npTEL/week7 : more generalRollNos.dat
190050021 65
190050010 42
197050015 78
190060037 91
180030001 88
190370051 91
180020041 66
190010045 44
190030101 83
190120077 91
~/Desktop/npTEL/week7 : ./a.out
190370051
91
190380051
Roll number not found.
-1
~/Desktop/npTEL/week7 : %
emacs -nw highest.cpp

[1]+ Stopped                  emacs -nw highest.cpp
~/Desktop/npTEL/week7 : s++ 3poly.cpp
+ g++ 3poly.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/
simplecpp/include -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/npTEL/week7 : ./a.out
~/Desktop/npTEL/week7 :

```


So, this is the programme. This is what you have on the slides. So, let us compile it. And let us run it. So, you can see that the 3 turtles are drawing 3 octagons, but they are drawing octagons in different positions. And this way, you can create interesting designs or interesting patterns, or you can choreograph the turtles in interesting ways if you want.

And incidentally you will remember that we did something like this earlier, when we were studying chapter 5. Over there we just had names t2, t3. But now, we can use an array. And since we have an array, we do not have to write the commands separately, for separate turtles, we can put them in the loop. So, this is as far as I wanted to say about arrays and the application of arrays. So, let me make some concluding remarks for this entire lecture sequence.

(Refer Slide Time: 7:55)

Arrays: concluding remarks

- Way to store many objects of the same type in memory, without having to separately define a variable for each.
- Index: used to choose an element.
 - Must be at least zero and at most array length - 1.
 - Can be an expression
- Index may sometimes play an active role:
 - When roll numbers are consecutive from 0 to N-1
 - Polynomial representation: ith coefficient stored in ith element
 - Taxi dispatch: indicates arrival order
- Index may not have significance:
 - When roll numbers took on arbitrary values
- Indexing into an array happens very fast, independent of how many elements are present in the array.



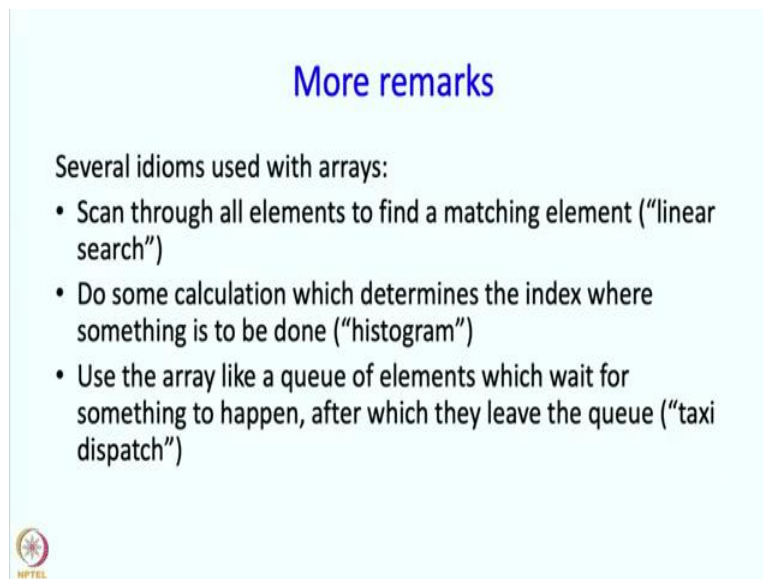
So, first of all, an array is just a way to store many objects of same types in memory, without having to define a separate variable for each object. So by defining an array, you get many variables defined in one shot. So, you do get all the variables you want, but you do not need to define them one at a time separately

The index allows you to choose which of the variables you want to refer to at the current movement. And the index must be between 0 and the length minus 1 or the size minus 1 the size of the array minus 1 and it can be an expression. And index may sometimes play an active role, in the sense that, when the roll numbers are consecutive from 0 to N minus 1, we do not need to store the roll numbers. So, the index or the position of the element is going to tell you what roll number it is. So, we do not store the roll number, we do not store the index either. But we can refer it to the index we can say, give me the ith element of this array, and that i is the index and that is implicitly present. And when we are dealing with certain kinds of problems. So, if the case of the roll numbers, if the roll numbers are in 0 to N minus 1, we do not really need to store the roll numbers. Similarly, in the polynomial representation, we had n coefficients. But we do not need to store i, we just have to store c_i , the ith coefficient. Because the coefficients are stored in sequence, we know which coefficient corresponds to which power in the polynomial. In the taxi dispatch, the index also played an active role, because using that and using that variable front, we could figure out which taxi driver came earlier and which taxi driver came later.

Sometimes, the index may not have any significance. So, in our example where we had general roll numbers as well as marks, so we had to store the general variables as well. And

essentially what was going on here, is that, we were storing sets of pairs and those pairs could have been stored in an arbitrary order. One point that we have to note is that, indexing into an array happens very fast, independent of how many elements there are present in the array. So, this is something that we will discuss a little bit later, in the next lecture, but this is something that you have to keep in mind. So, when you write, `marks[i]`, little bit extra time is there beyond accessing a simple variable. But it is almost the same time accessing an ordinary variable, just a little bit more. And it does not depend on how many elements you have in the array.


(Refer Slide Time: 11:26)



More remarks

Several idioms used with arrays:

- Scan through all elements to find a matching element (“linear search”)
- Do some calculation which determines the index where something is to be done (“histogram”)
- Use the array like a queue of elements which wait for something to happen, after which they leave the queue (“taxi dispatch”)



Some more remarks, when you have arrays, there are some natural idioms that you should understand. You may have to go through all the elements in the array, to find the matching element. So, this is called a linear search. You may do some calculation, which determines the index of the histogram., index of which element is to be looked at. So, you may not go through indices in order always. But you may do some interesting calculation. And this is something that happen in the histogram example. You may use an array like a queue of elements, which wait for something to happen, after which they leave the queue. And this is something that happened in the taxi dispatch example. And of course we also saw the accumulation idiom, we also saw the filtering idiom. So, that is it for this lecture. I will remind you that in the textbook, there are lots of problems related to arrays, at the end of chapter 14. And I would like you to start solving those problems. But we are going to have an additional lecture on chapter 14, which is going to come up next. So let me conclude this lecture at this point and thank you.