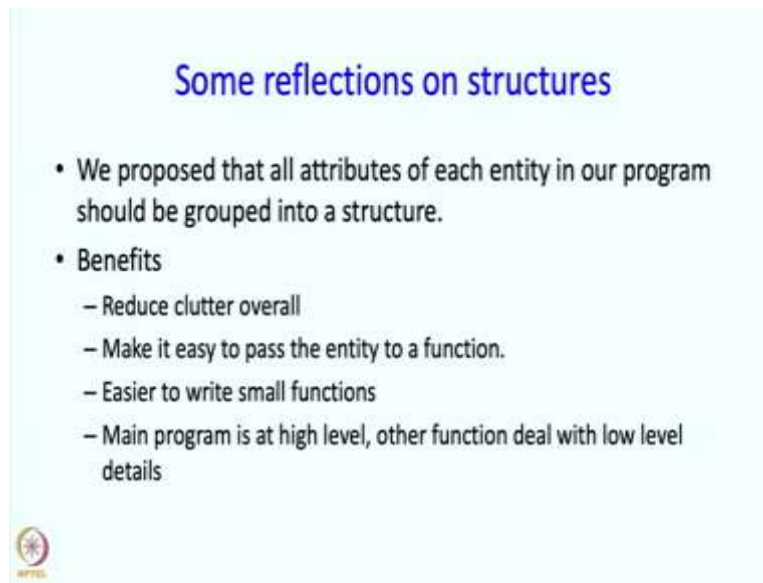


An Introduction to Programming through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Lecture 20

Part-1 Introduction to Member Functions


Hello and welcome to the NPTEL course on an introduction to programming through C++ I am Abhiram Ranade and in this lecture I will continue to talk about structures. So, some reflections on what we have seen so far.

(Refer Slide Time 00:36)



Some reflections on structures

- We proposed that all attributes of each entity in our program should be grouped into a structure.
- Benefits
 - Reduce clutter overall
 - Make it easy to pass the entity to a function.
 - Easier to write small functions
 - Main program is at high level, other function deal with low level details



So, we proposed that all attributes of each entity in our program should be grouped into a structure and we said that the benefits for this are that it would reduce the clutter overall it will make it easy to pass the entity to a function, we can pass it as a single argument rather than all the different variables which are associated and because we have fewer arguments, we will be more likely to write lots of functions, and also each function will have its concern very limited. And therefore understanding what is going on in the function and reasoning about it in general will be easier to do.


(Refer Slide Time: 01:24)

An observation

- Typically the attributes of an entity are accessed in only a few different ways.

Example 1: Vector from physics:
Example 2: Queue in taxi dispatch

- Both vectors and queues will have many attributes/members.
- We do **not** read/update the individual members in isolation.
- A natural operation on vectors or queues requires us to simultaneously access/update several attributes/members in a very specific manner.



Now here is an observation about how such entities are processed. So typically, there may be many attributes that an entity might have but they are accessed in only a few different ways. So I am going to take two examples, one is a vector from physics and another is the queue that we had in taxi dispatch problem that we saw earlier.


So in both these cases the vector and the queue will have many attributes or many members, but you will note that usually we do to read or update the individual members in isolation. A natural operation requires us to simultaneously access or update several attributes, and in a very specific manner. So what does it mean? So it means really, that we should support this kind of access and update rather than the update involving individual members.

(Refer Slide Time: 02:56)

Vector from physics

- A vector may be used to represent velocities, positions, displacements, electric fields, ...
- A vector can be indicated by its components in x, y, z directions.
- So we may represent a vector as

```
struct V3{ double x, y, z;};
```
- There could be other representations, e.g. polar.
- Key observation:
 - You will rarely read just the x component, or update the y component.
 - Most likely you will do something with all 3, e.g. add two vectors, scale a vector.

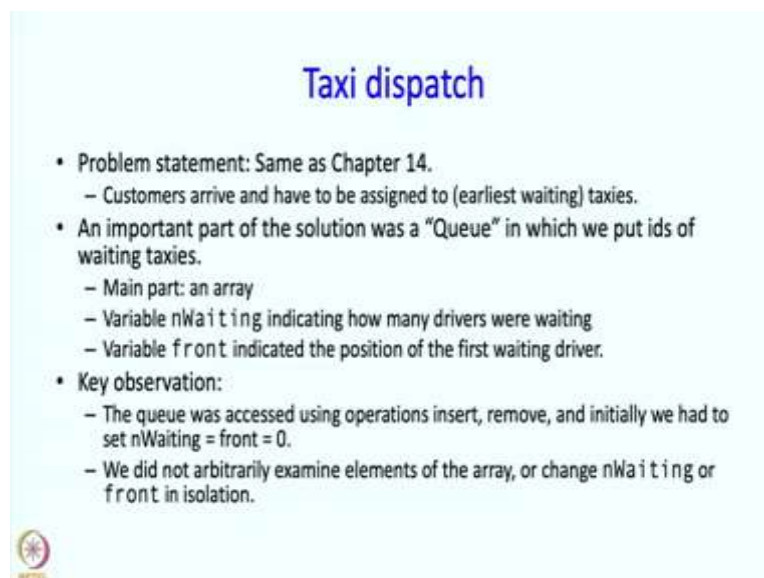


So let me show these actually these examples little in more detail. So, vector from physics so in physics vectors are used to represent lots of things velocities, positions, displacements, electric fields and say if it is a vector in three dimensions, then it can be indicated by its components in the x,y,z directions. So maybe we might have a vector type V3 and it contains members x, y, and z.

There could be other representation as well, I mean there could be the polar representation for example or the cylindrical coordinate representation. So, here is a key observation so for this representation you will rarely read just the x component or rarely update just the y component. What you typically do is, will involve something with will involve all three coordinates so maybe we will add two vectors together or maybe we will scale up a vector.


So both these operations will involve changing all the coordinates and the change will happen in the very specific manner.

(Refer Slide Time: 04:05)



Taxi dispatch

- Problem statement: Same as Chapter 14.
 - Customers arrive and have to be assigned to (earliest waiting) taxis.
- An important part of the solution was a "Queue" in which we put ids of waiting taxis.
 - Main part: an array
 - Variable `nWaiting` indicating how many drivers were waiting
 - Variable `front` indicated the position of the first waiting driver.
- Key observation:
 - The queue was accessed using operations `insert`, `remove`, and initially we had to set `nWaiting = front = 0`.
 - We did not arbitrarily examine elements of the array, or change `nWaiting` or `front` in isolation.



The taxi dispatch problem we saw earlier, and it is discussed in chapter 14 of the book. So briefly we have customers arriving and have to be assigned to waiting taxis, and important part of that solution was a queue, and into this queue we put IDs of the waiting taxis or rather the IDs of the waiting drivers and the main part of the queue was an array and these elements where the one in which we store the IDs, and then there was a variable `nwaiting` indicating how many drivers were waiting and variable `front` which indicated the position of the first waiting driver.

And again, we can make the observation that the queue was accessed using operations insert remove and initially we had to set $n_{\text{waiting}} = \text{front} = 0$. We did not arbitrarily examine the elements of the array or change n_{waiting} or front in isolation. So whenever we make the change there was some high level operations that we had in mind and over code was implementing that high level operation.

(Refer Slide Time: 05:38)



The insight

- We should put all the attributes of an entity into a structure, and,
- We should provide functions with which to access the attributes.

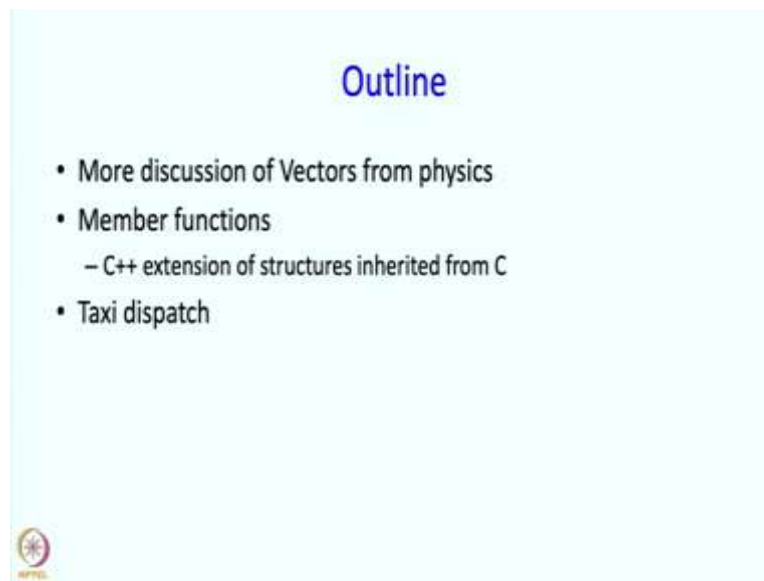
"Member functions" : Main topic of this lecture

- We should avoid accessing attributes directly
 - Very likely this indicates an operation that has not been clearly thought out, and which might lead to errors.
- We must think about and catalogue "the right ways of accessing an entity".

So the insight from all of this is that first of all we should put the all the attributes of an entity into a structure but in addition, we should provide functions with which to access the attributes. And these member functions these functions are customarily member functions or the functionality that we want for accessing is provided by something called member functions and these member functions are the main topic of this lecture.

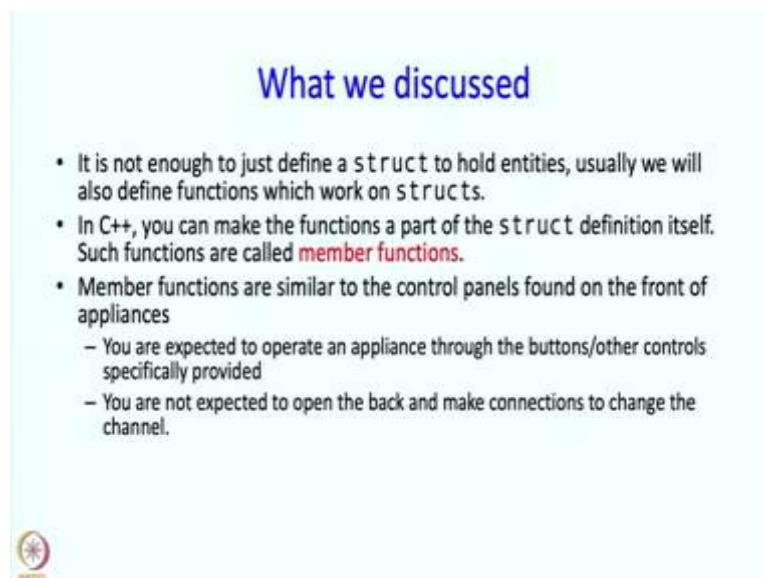
So to put it differently we should avoid accessing attributes directly, why is that, well very likely this indicates that we are doing an operation that is not been clearly thought through and therefore may be it contains errors. So, clearly something real or even abstract, but sort of something that we clearly know, something that we clearly understand like a queue or a vector has its own logic and that logic has to be incorporated into functions and we are going to describe this notion of member functions which is going to be most useful for incorporating such logic. So we think about and catalogue the right ways of accessing an entity.

(Refer Slide Time: 07:09)



So here is the outline of this lecture, we are going to have a bit more discussion on the vectors from physics then we will introduce this notion of member functions and I should mention here that member functions were not present in C, were an extension to what we inherited from C and the extension is first proposed and implemented in C++. So, this is kind of a more modern feature if you will. After discussing member functions, we will talk about taxi dispatch and see how that as well can be written using member functions. Alright, so what did we discussed in this part of the lecture.

(Refer Slide Time: 08:08)



So we said that it is not enough to just define a struct to hold entities. Usually we will also want functions which work on the structs in C++ we can make the functions a part of the

struct definition itself and such functions are called member functions. In some ways, member functions are similar to control panels found on the front of appliances. So, you are expected to operate an appliance through the buttons or other controls that have been specifically provided. You are not expected to innovate or you are not expected to do tricky things when you operate an appliance say for example you are not expected to open the back and change some capacitors or change some wirings.

So if you want to change the channel on a television there is a very very clear way to do that and analogously, the thought is that member functions indicate to you what is sort of the clear proper way of doing things and in some sense you do not want users to go looking into the data members themselves and changing the data members because they might be making mistakes in doing that. So next, we are going to develop this idea and for that we are going to look at the vectors from physics but, let us take a quick break.