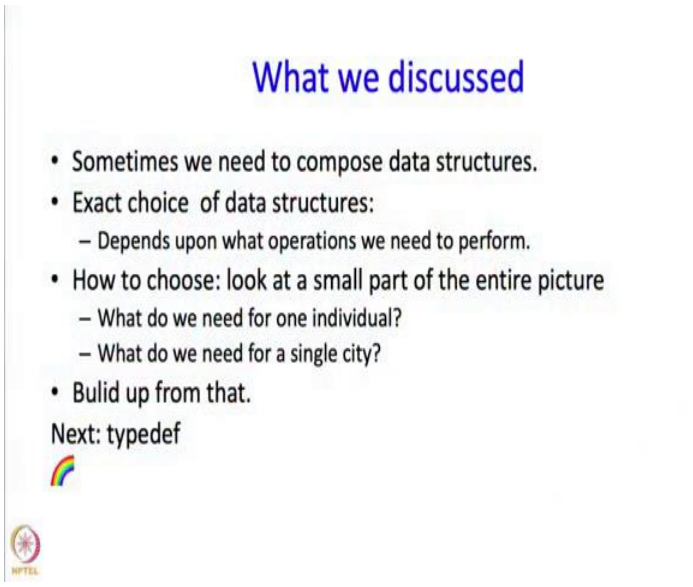
An Introduction to Programming through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Lecture No. 24 Part – 5
Data structure based programming
Typedef and lecture conclusion

Hello and welcome back.

(Refer Slide Time: 0:25)



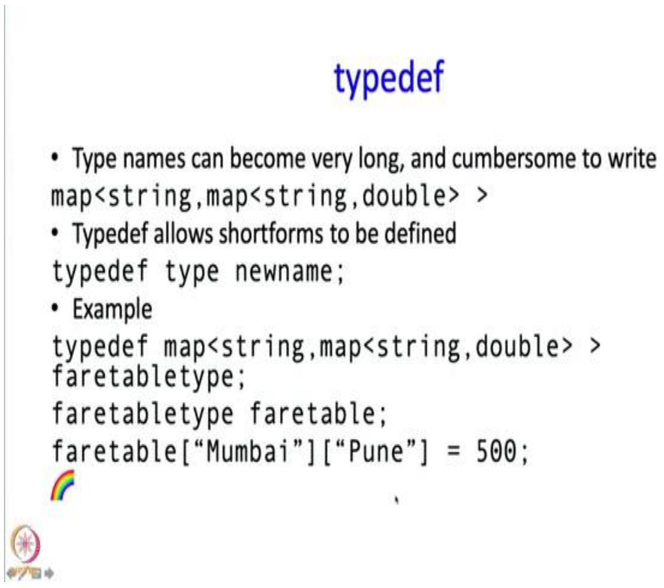
What we discussed

- Sometimes we need to compose data structures.
- Exact choice of data structures:
  - Depends upon what operations we need to perform.
- How to choose: look at a small part of the entire picture
  - What do we need for one individual?
  - What do we need for a single city?
- Bulid up from that.

Next: typedef

In the previous segment we talked about how we can compose maps, and vectors and strings together to build interesting data structures. In this segment, I am going to talk about somewhat technical, somewhat syntactic called the typedef statement. And then I am going to conclude this sequence of segments.

(Refer Slide Time: 0:45)



typedef

- Type names can become very long, and cumbersome to write

```
map<string,map<string,double> >
```

- Typedef allows shortforms to be defined

```
typedef type newname;
```

- Example

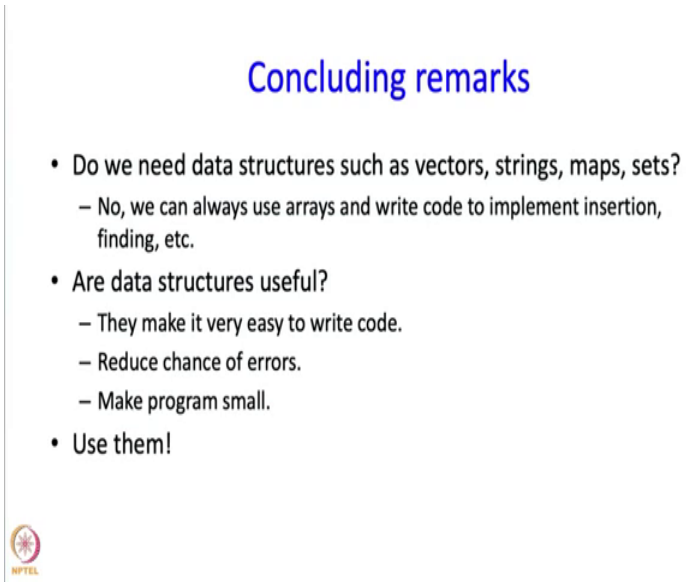
```
typedef map<string,map<string,double> >
faretabletype;
faretabletype faretable;
faretable["Mumbai"]["Pune"] = 500;
```

So, typedef, so let me first observe that type names can be become very long, and cumbersome to write. So, for example, we have already seen that that fare table that we constructed had this type. So, we have to write it when we define that variable, but now suppose I want to pass that variable to a function, then even there we will have to type this as the type of that argument. So, you may think that look typing this again and again is very cumbersome and also it does not really tell me what this is all about.

So, basically the typedef statement allows short forms to be defined and the idea is this, so I am, I write typedef, then I want I write the type for which I want to create a short form and then this is the short form. So, for this, I could have write some, I could have written something like this. I would say typedef and this is the type, this very type mentioned over here and for that I am going to create the name called fare table type. So this is the type and this is the name that has been created this is synonym for this long type name.

And how do I use it? Well, exactly as you think, I can the write faretabl type, fare table so this creates a variable named fare table of this type. Actually, in the last example, I called it fare, but since, it really is a fare table maybe I thought I should use this full name. So, just to complete, I could write faretable[Mumbai][Pune] equal to 500 or cout fare table Mumbai, Pune. So that concludes discussion about typedef.

(Refer Slide Time: 2:48)



Concluding remarks

- Do we need data structures such as vectors, strings, maps, sets?
  - No, we can always use arrays and write code to implement insertion, finding, etc.
- Are data structures useful?
  - They make it very easy to write code.
  - Reduce chance of errors.
  - Make program small.
- Use them!

So, let me now make some concluding remarks about this sequence of segments. So, let me ask the question to you. Do we need data structures such as vectors, strings, maps, sets or could we live without them? Well we can leave without them. So, they are not strictly needed, in fact, the language like C does not have these data structures. And what is done there? Well, the programmer has to really use the arrays and write code to implement insertion, finding, etc.

Or some other kinds of constructs that the programmers himself or herself has to create. So, these data structures or these classes are strictly not necessary. But are the useful? You bet! They are amazing! They make it very easy to write code. So, the designers of these data structures have guessed and they have guessed very well, what kinds of things people want to do? And they have put the things into these data structures.

And further more these data structures are template classes. So, you can instantiate them with any type, so you can create the vector of ints, you can create a vector of doubles, you can create a vector of circles, you can create a vector of vectors. As you just saw so all of these things are possible. So, whatever type data structure you want that will get created. And it will reduce chance of errors.

So, because all these data structure has been created by expert programmers, and they have been tested, not just by those programmers, but because they have been released to the entire world, the entire world has tested these data structures for you. So, they are unlikely, it is unlikely that there is any error in them and they are so compact that using them is also very convenient and unlikely to produce errors.

And, of course, they make programmes small. So whatever your logic is you can see it often in a glance. That look this is what I am doing? Instead of having to go over several pages of code. So, it makes for improved program readability as well. So, definitely use them! Alright so that is the name message of programme and that is the end of this lecture. Thank you.