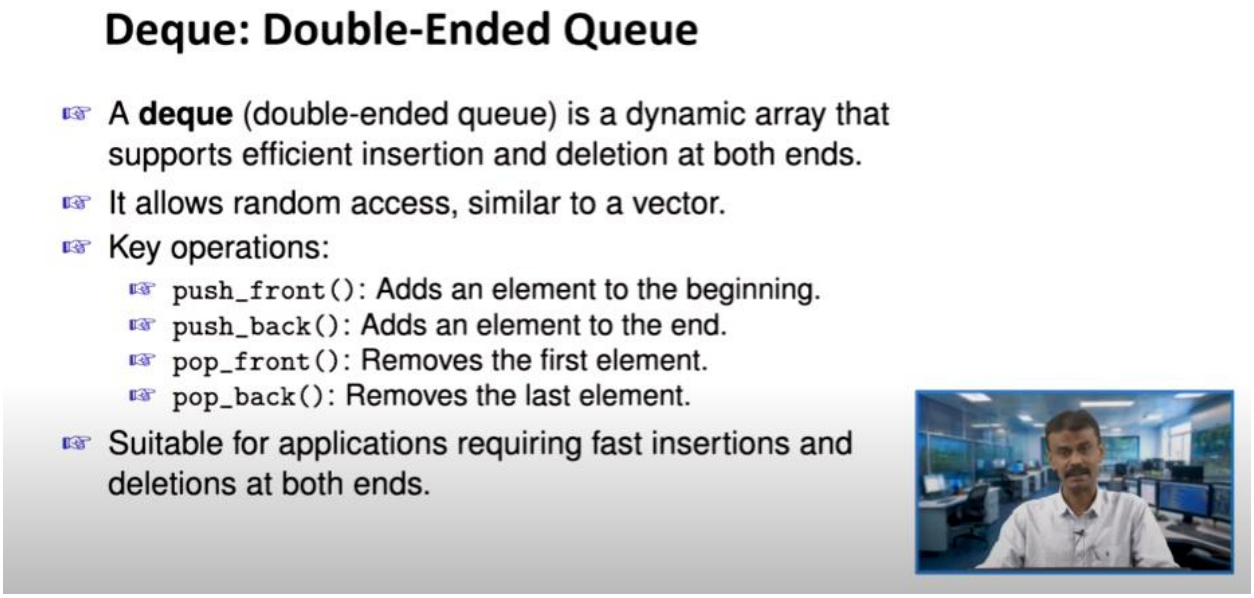
FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture42

Lecture 42: Associative Containers



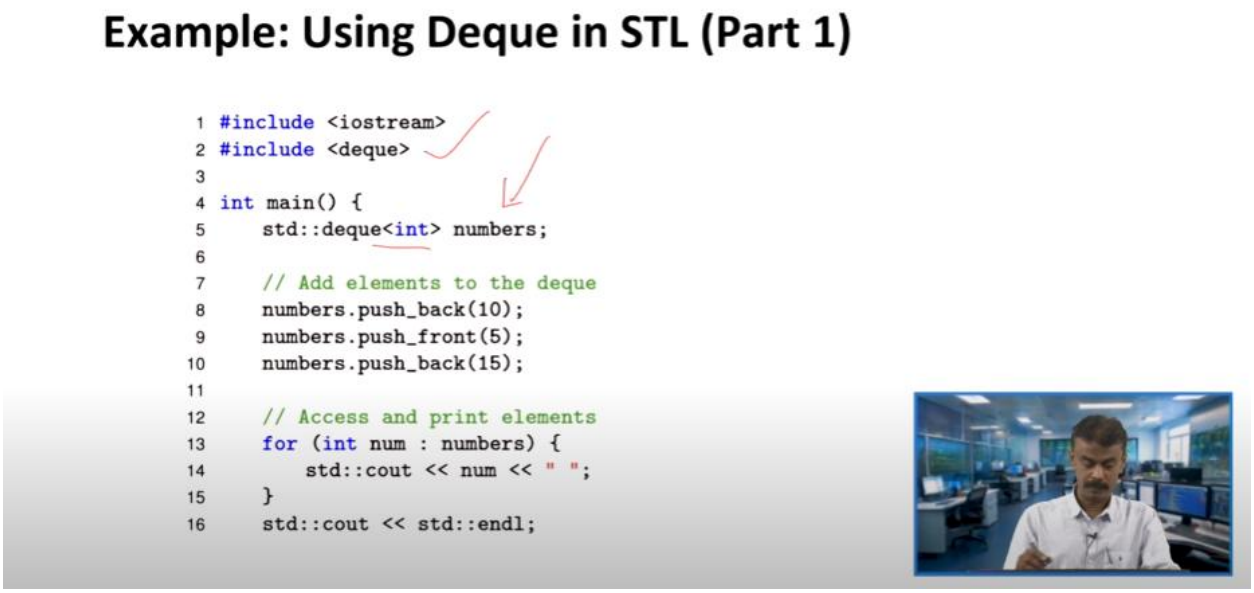
Welcome to lecture 42 standard template library we called as STL. In the last lecture, we talked about vector and list. If you recall, we have seen the programs also. And in this lecture, I will start with DEC. So, DEC is nothing but double ended queue or doubly ended queue.

So, this is a dynamic array where you can do the insertion and deletion at both ends. So, you can see many practical situations, right. So, where someone does not follow the rules, right. It is like they enter in the first, right and a genuine person enter in the last, right. So, that is nothing but doubly in that queue or deck.

So, this is also allowing the random access like exactly like a vector. So, we can have a random access over here right. So, it allows a random access and there are several operations we can do like vectors and lists. So, here we can have push front or insert front right. So, that means you can add an element in the beginning and push back right.

So, you can add an element at the end, right end of the queue, pop front, right, delete front. Right. So, that means you can remove the first element and pop back, delete back. Right. So, that means you are removing the last element.

So, in fact, when you want to have fast insertions, right, the practical applications, right, there are cases if you require fast insertions and deletions at both ends, this particular data structure is being used, and you call it as W and that Q or deque. So, now, we will take an example. What will I do? I will include deque, right.



Hash include iostream and hash include deque, and under deque class, it is an integer. We take numbers, all right. So, numbers is an integer; it is coming under deque, OK? Numbers is an object, numbers is an object coming under deque, the objects as the integers. So, now you can see, right. So, we have an advantage over here to take the member functions, right, inbuilt functions, inbuilt member functions.

So, the object 'numbers' is invoking push_back. Right, so when it is invoking push_back, this is initially an empty queue. So, we are inserting the first element, 10. Push_back or push_front doesn't matter because it is an empty queue, so 10 will be here, right? Next, 'numbers.push_front', so we are pushing 5 here. Right. So, pushing 5 here. I am not writing any pointer or something here.

Now, it is understood: 5, 10, it will be like this. Now, 'numbers' will invoke push_back by inserting 15. So, it will be like this. So, here you have 15, correct. So, now, if you look at the queue, right, it is a deque.

So, 5 is inserted in the front, and 15 is inserted in the back, and you can see. So, this is a deque, right. So, this is a deque. So, the usual way you do a queue is that the first element is 10, next you insert 5, and then 15. Your usual queue, we know that, right. So, usual queue if you take, right.

So, in the usual queue, what would have happened? It would be 10, 5, and 15, right. So, that is the usual one, but whereas in the case of a deck. So, this is possible. So, when we have the inbuilt functions like push_underscore_back. push_underscore_friend, right.

So, we can insert like this. So, now we have to forget this. So, our deck is nothing but this. So, this is our deck, right. So, now we are printing, right.

So, the numbers, the object base address now num will point, and when you are printing num, correct. So, it is an integer, correct. So, this is an integer num, and the address of num will point to the base address numbers, and then when you are printing number. All the elements will be printed, right? So, 5, 10, 15, these will be printed, right?

So, we will continue. You will have one end L, that means a new line. So, now I am doing pop front, right? So, when I do pop front, so what will happen? So, here you go pop front, right?

That means I delete the front element. So, when I delete the front element, this will go off, So, that is a meaning pop friend for line number 18 and line number 19 pop back. So, that means I am deleting 15 as well correct pop back delete the last element. Now, I print num right.

So, the numbers the base address is assigned to the address of number and then when you are printing the number. So, you will get this particular output. So, that means we are getting 10. right and then you have endl return 0 what we can do we will check whether we are getting this output so i will just go to c++ compiler so this is an example right the same i have explained line number 19 you see pop front pop back and yeah the same program let us run the code

So, when you run the code, you can see the output right? What you are getting is 5, 10, 15, and then 10, right? So, developer, in C++, we are running and you are getting the output. So, this is how the doubly-ended queue works, right? So, the doubly-ended queue works. So, when you are studying a data structure, you have to write the code: pop front and pop back, right? So, the advantage of STL here is, once you are including deque, right?

So, you have the option to use the inbuilt functions under deque: push back and push front, right? So, this is how the doubly-ended queue is working. So, next. What do you mean by associative containers? Right.

So, associative containers—what it will happen—like set. Right. So, mathematically, when you define a set, assume that when A is equal to, let us say, 3, 2, 3, 4, 5, and 5. Right. So, we know that when I am representing this.

Right. A like let us say 3, 2, 4, 5. am still right right because the elements are repeating 5 is repeating 3 is repeating and we know the basic mathematical definition of a set it will be unique and in the case of associative containers so there is some property it will write it in sorted order right so in fact so we will get right in the case of associative containers we will get 2 3 4 5 this is also same Both are same, all are same, right?

But this will follow the associative containers in this C++ follow this, all right? And also, so with the help of keys, I can easily retrieve what are all those numbers, all right? And in fact, it is having some ordering base. This is what I explained in the set notation. So, it is using some ordered base.

What operator is being used? The less than operator is being used. So based on this, for example, initially I gave 3, 2, 3, 4, double 5. Right. But it will keep like 2, 3, 4, 5.

That is the meaning of the second point. Right. Internal ordering. Right. So, associative containers.

So, we can classify them like set, multiset, map, and multimap. So, a set stores unique keys in sorted order. You can see the unique keys: 2, 3, 4, 5. And they are in sorted order. This is an example.

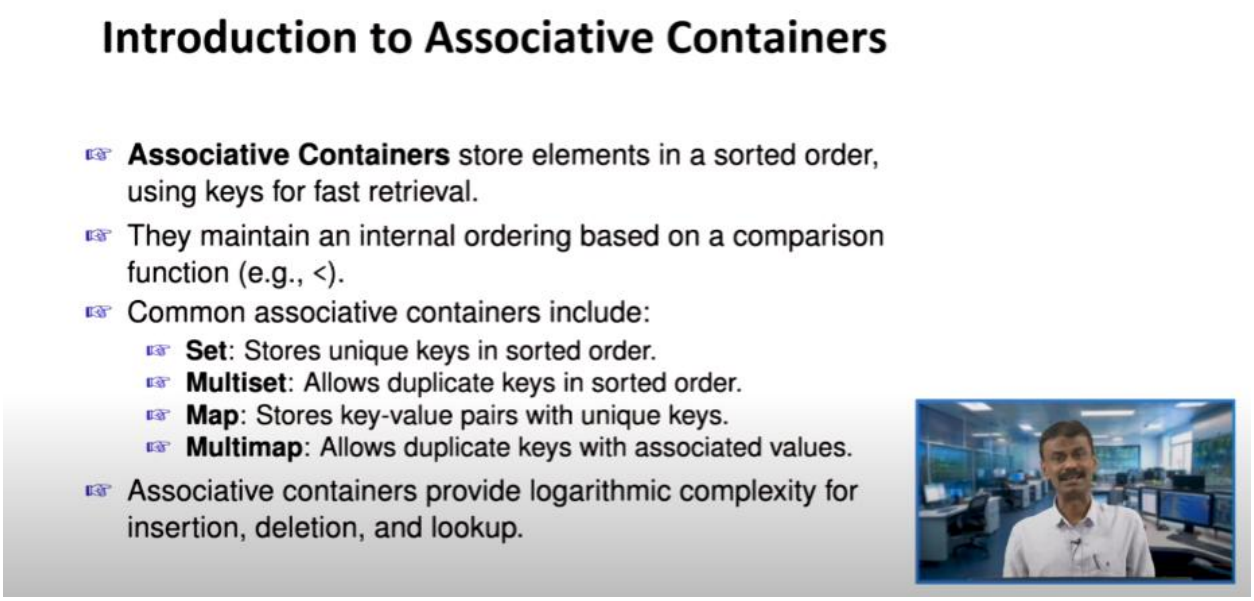
Whereas, a multiset allows duplicate keys also. Alright. For example, in this particular case, when I write it as a multiset. So, A will be equal to. Right.

I am explaining the multi set. A will be equal to 2, 3, 4, 5. That is the meaning. Alright. So it is also allowing duplicate key.

You can see. 3 are duplicate. 5 are duplicate. And then it will be in sorted order. That is the meaning.

All right. I hope you understand. Then map. So map shows key value pairs. So you can have a pair.

Right. Let us say 2, 3 or bala, 50. So one is the string, another is the age. Right. So it will keep the pair, and you can have the unique keys.

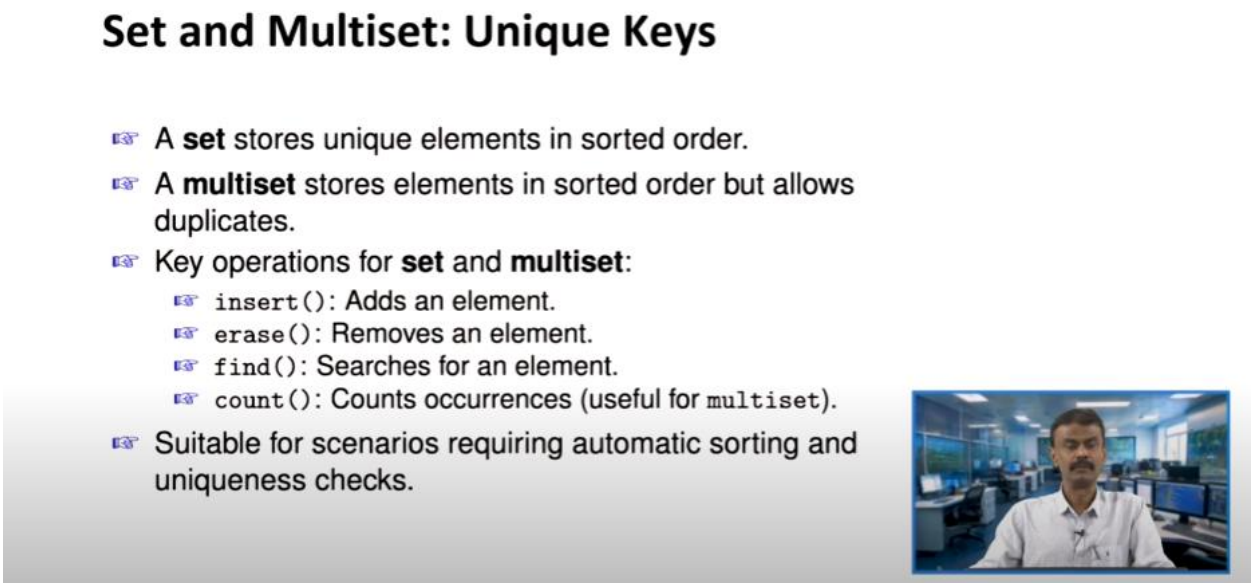


Right. So you can have the unique keys. So when you are representing, you will have the unique keys. So, unique in the sense, for example, when you are joining the institute, college, or university, you will be given the enrollment number, which is unique, right? So, it will have the pair of values, right? The pair, in fact, is a pair that is a value to—I mean, let us say two values—and it will have the unique keys concept as well. It will have unique keys, and the next one is a multi-map.

So, here the duplicates are also allowed, right? It is allowing duplicate keys with associated values. So, when I am using associative containers, what is the advantage, all right? So, I can do the dictionary operations like insertion, deletion, lookup, etcetera,

in logarithmic complexity, right? So, those who have studied or those who are going to study data structures or algorithms, this is the order log n, right? N number of elements.

You can retrieve, insert, delete, look up, or update in order log n, right? That means log n steps, where n is the number of elements in the set, multiset, map, or multimap. The number of elements, all right. And you can do all these basic dictionary operations in order log n time, which means approximately log n steps. So, as we said, right? What do you mean by a set that stores unique elements in sorted order? We will see an example, and similarly, a multiset.



So, that allows the sorted order, but the only difference is duplicates exist, right? What are all the key operations that we can do? Insertion, erase, find, count. So, these are all the inbuilt functions. So, insertion means you can add the element, and again, it will get sorted. Erase will remove the element.

Find will search for an element. Count will give the number of elements in the set or the number of elements in the multiset, right? So, count the occurrences. So, particularly when you use a multiset, because I talked about duplicate elements. So, there it will be useful, and the main applications are, I mean, you can have automatic sorting and also check for uniqueness. Let us take an example, right.

Example: Using Set

```
#include <iostream>
#include <set>

int main() {
    // Using set
    std::set<int> uniqueNumbers = {10, 20, 30, 20};
    uniqueNumbers.insert(40);

    // Print elements
    for (int num : uniqueNumbers) {
        std::cout << num << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

Output

```
10 20 30 40
```

So, you will understand much better. So, let us include set that is another advantage right hash include set. So, now you use unique numbers under set right which is our set of integers right. So, those unique numbers it is under set you take 10, 20, 30, 20. So, I am using the duplicate elements also right duplicate element.

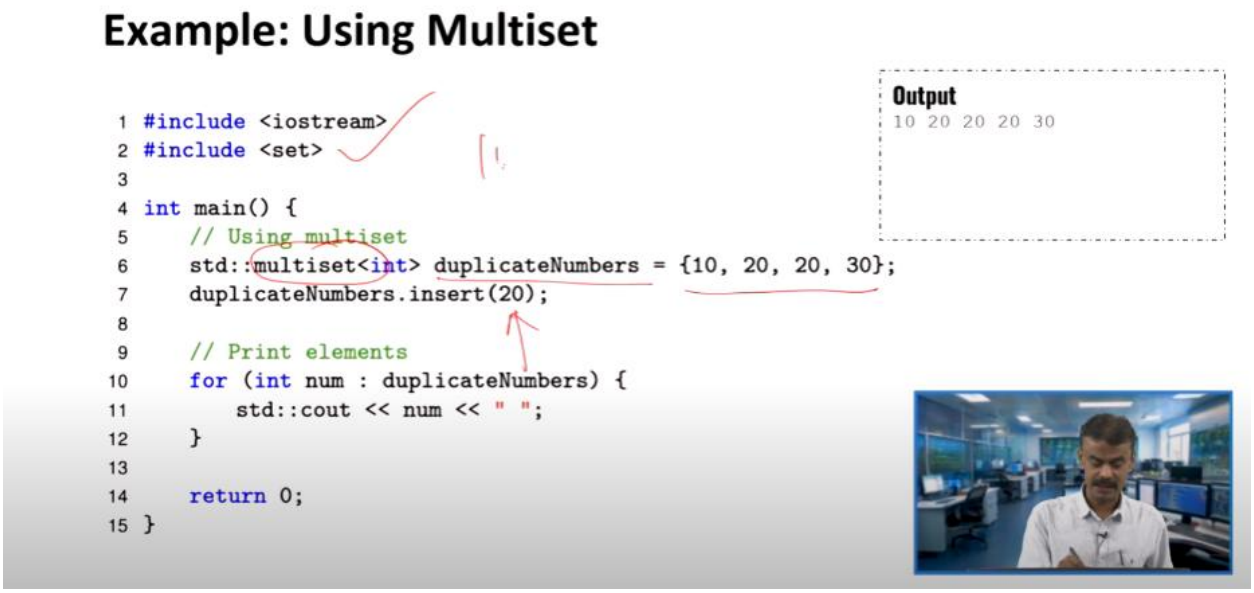
So, this is my initial set and then I want to insert. So, I have included set right. So, there is no duplicate and it will be in ascending order this is what we studied. So, what we are expecting? So, even though you have defined this you will have 10, 20, 30 right.

So, you will have 10, 20, 30 and then you are inserting 40 right the unique numbers object right which is invoking insert member function inbuilt function under set and then you are inserting the element 40 right. So, when you are inserting an element 40, 10, 20, 30, 40, all right, and we know. So, this will be in ascending order. So, unique numbers, correct, this base address will assign to the address of number. And when I want to print the number, right, it is nothing but star unique numbers of, in fact, unique numbers of 0, 1, 2, 3, etc.

So, that will be assigned to number. This we had seen in the last program. So, number, so 10, 20, 30 will be printed. all right. So, in fact, when I run the code and then endl, it will go to the next line.

So, in fact, when you run the code, you can see the output 10, 20, 30, 40. So, we can check the code also. So, using set, I will just go to the C++ compiler, same example 10,

20, 30, 20 and then I am inserting 40, all right. So, when I run the code, you can see the output, yeah. So, 10, 20, 30, 40 as the output.



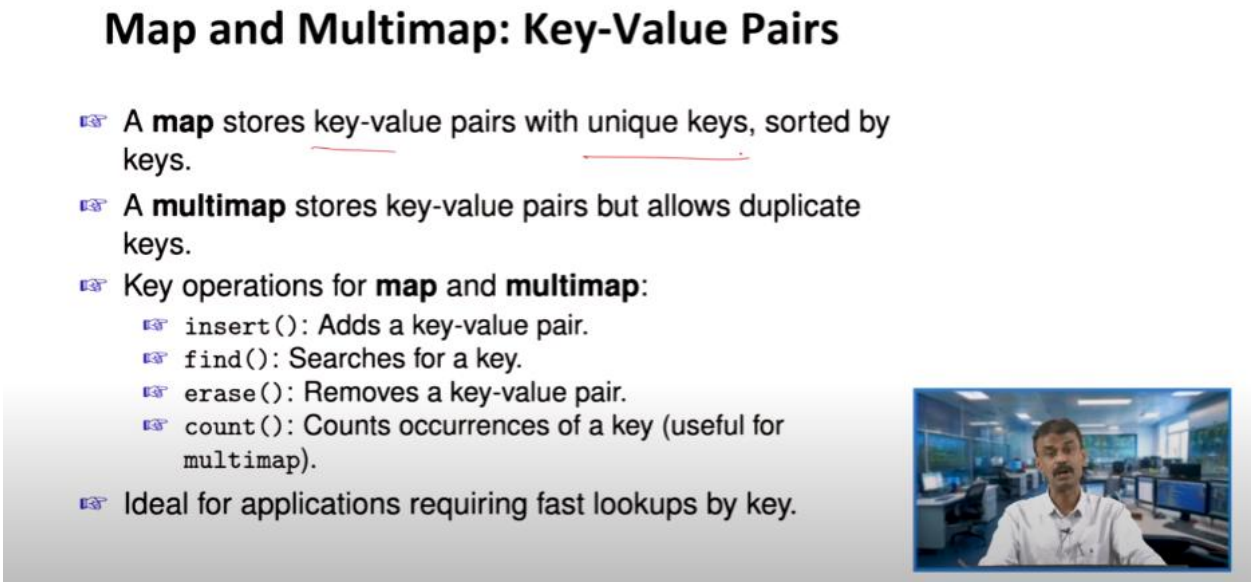
So, you practice, right, you include all these header files and you can keep on practicing. So, we had seen several write STL and set is one of them right. So, anyway I have given the detail explanation all right. So, the set we know that it stores a unique element and it is in ascending order and duplicates yes it will be ignored because 20 is appearing two times right first index and third index.

So, it is being ignored. And the loop is iterating to all unique elements. So, in fact, we got the output the unique elements. So, there are scenarios all right where duplicates are not allowed all right. So, you cannot enter the same mobile number for two different persons that is a practical scenario or you cannot assign same enrollment number for two persons correct.

So, these are all the scenarios where you can use duplicates. the set and the next example is a multi set all right. So, what we have studied here duplicate exists even if duplicate exists the elements will be in sorted. So, again you are including set I define duplicate numbers under multi set class multi set class of integers duplicate numbers is a object. So, that contains 10 20 20 30 duplicate exists 20 is a duplicate

Now, I am again inserting 20. So, the set contains 10, 20, 20, 30 and then I am inserting 20. So, when I want to insert 20, what will happen? I will get the output like this. I should get the output like this 10, 20, 20, again 120, then 30.

The reason is it has to be in the sorted fashion. So, after inserting 20 it will be in the sorted fashion and when you are printing this correct. So, duplicate numbers object you have the corresponding address in number and the number when you are printing right. So, this particular for loop you are getting this is the output correct. So, in fact, when I run the code I will get the output like this.



So, it is allowing duplicate elements all right. So, in this case it is a multi set is allowing duplicate elements that is the difference The previous case and only set we use only set right that is the difference between set and multi set. So, year 20 was appearing multiple times, but still it was working all right. So, and of course, the loop iterates through all the elements including the duplicates and there are scenarios where the duplicate exists.

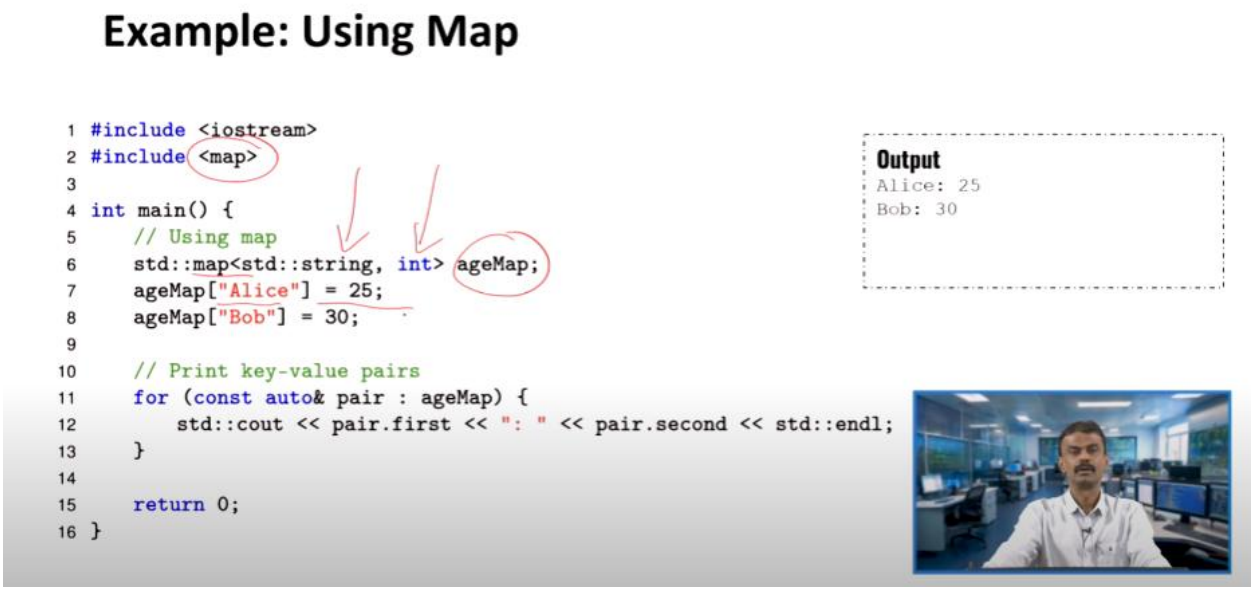
So, at that time what you can do it is advisable to use multi set. The next concept map and multi map. So key value pairs. So map as I said. So it will have key value pairs.

So you will have a pair. Maybe the name and age. Name is a string. And age is an integer. And then you can find the unique key.

And the unique keys are in sorted fashion. And another thing: multi-map. It is also a map. But it allows duplicates. Like exactly how we define certain multi-sector.

So multi-map also allows duplicates. Like a multi-sector. So, it is a key-value pair. So, here you have several operations under map and multi-map: you have insert, you have find, right? So, insert will add a key-value pair, find will search for a key, and erase will remove a key pair, right? It is like a delete. Count the occurrences of a key, all right.

So, since duplicates are allowed in a multi-map. So, this particular function count which counts the occurrences of a key, which will be useful, right? So, when you have a lookup table, there are many applications where you have lookups. So, in those particular applications, when you are using a map or multi-map.



So, this will be useful. So, let us consider using a map, right? That means I am including a map apart from your usual IO stream. Right. So here, I have age_map as an object whose class is map.

So the map contains a string and an integer. Previously, we had seen only integers or, in fact, only one data type. Right. Set and multiset. So here, if you see, we have a pair.

One is a string. Another one is an integer. One data type is a string. Another data type is an integer. And the age map is an object.

So now we are writing the age map of one string name, right? Alice, right? So the age map of one string name Alice is equal to 25, all right? So let us say this ID or age, whatever. Yeah, it's in fact the age map, right?

So age, 25 is an age. And line number 8, you have the age map of Bob, all right? The age map of Bob. So, that is equal to 30, right? So, which is equal to 30, fine.

So, now these two are being stored. So, now for const auto address right. So, const auto address you are having pair right. So, you are having pair. So, age map that means the address of age map and the address of pair right.

So, they are pointing to the same location that is a meaning over here right. So, when I write rn equal to let us say n equal to 10, int address rn equal to n, all right. So, meaning is address rn and address n, they are pointing to the same location, right. So, in a similar way, auto address pair colon age map.

So, the both are pointing to the same location, right. So, now, based on this, if I put cout, pair dot first and there is one colon pair dot second, So, pair dot first is nothing but whatever be the string. What is the string? Allies.

Pair dot second, it is nothing but an integer. What is the integer? 25. So, the first output I will get is a for loop. Alright.

So, it will move. Once it is printed, it moves to the next address location. So, pair dot first will be printed, followed by a colon, and then you have pair dot second, which is 25. Alright. So, then next.

Right. It is moving to the next for loop. Moving to the next location. So, the next location now has pair dot first. What is that?

Bob, right? Followed by the colon and then pair dot second, the age is 30. All right. So these two will be printed. Allies colon 25 and a Bob colon 30 will be printed.

So now we will run the code, right? Because map we are seeing. first time. So, what we can do we will go to the C++ compiler and if you look at here the map the same code and if I run the code compile and run the code. So, you will get the output right alice colon 25 and bob colon 30 right.

So, this is the output we are getting. So, this is the case of map. All right. So, we will see what do you mean by multi map. All right.

And this is the explanation I have already given the map that is storing the key value pair. So, you can see that the key value pairs are nothing but allies and Bob. All right. And then you have 25 and 30. Right.

And the keys are stored automatically. Yes, we have seen that when you are printing out this first and second. all right and if you want to access right so i want to find out let us say i want to print what's bob's age in that particular example or you have n number of keys assume that you have n number of key pass so if i want to access any one so it will take approximately login steps and similarly i want to insert correct so map that follows a sorting order right so it will be as i said it will be sorted automatically So, when I am inserting of course, it has to be sorted.

So, even in this case also approximately it will take log n times all right and here we are using the auto keyword all right. So, automatically it will be taking care of the data type auto in C++ automatically taking care. So, for example, here we have used string and integers. right. So, if you use auto, auto can behave one case pair dot first it is behaving for string right or it is mapping to string and pair dot second it is mapping to integer that is the meaning.

So, that is the idea of using auto; you can try it, right? So, auto a equals 10, auto b equals 9.8, right? So, in fact, you try to print or try to find out the size. So, automatically, it will tell you the size of an integer, the size of a double or float, right? So, you can write for the float also.

Example: Using Multimap

```
1  #include <iostream>
2  #include <map>
3
4  int main() {
5      // Using multimap
6      std::multimap<std::string, int> multiAgeMap;
7      multiAgeMap.insert({"Alice", 25});
8      multiAgeMap.insert({"Alice", 28});
9
10     // Print key-value pairs
11     for (const auto& pair : multiAgeMap) {
12         std::cout << pair.first << ": " << pair.second << std::endl;
13     }
14
15     return 0;
16 }
```

Output

```
Alice: 25
Alice: 28
```

So, you can try with the auto keyword. So, the next concept is a multimap. So, multimap, as I said, duplicates are allowed, right? So, duplicates are allowed; what we can do here is the same or similar code, right? So, you have a multimap.

So, here I am having The object multimap, and then you have the object multimap, all right. So, which has string and integer, all right. So, first, initially, multimap is invoking insert; you have allies and 25, right? Allies, comma, 25. In the previous case, we have done slightly different. I mean, this is also a way you can do it; I am calling the insert function, right? The insert member function.

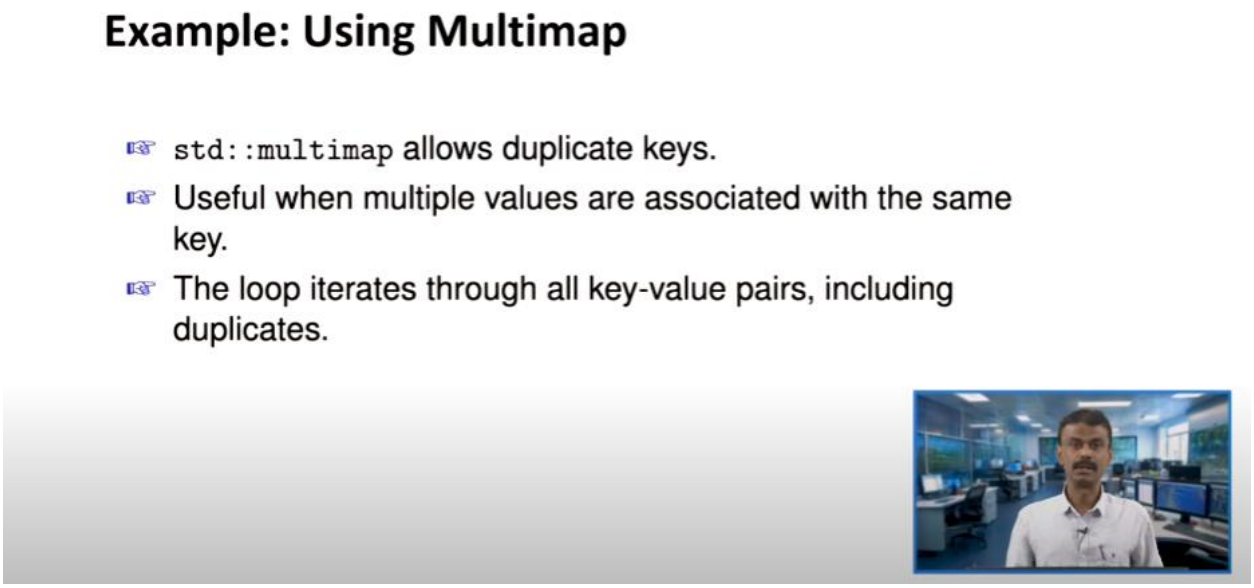
So, inbuilt function allies and 25. And line number 8, you have the same multi-age map object is invoking insert. Now, you are putting allies and 28, right. If you do the previous case, map case, I mean it will be overwritten, kind of, right. The duplicates are not alone, right.

Whereas, here allies, again you are having the name allies. So, what it will be doing? The same we will do, auto, right. Reference object is pair. So, multi-age map address will be now same as pair.

Right. Address pair. And then when I am printing pair dot first colon pair dot second. Right. So for is initially pointing to allies and 25.

Correct. So, allies followed by the colon and 25 will be printed. Right. And you can see the second insertion was allies comma 28. So, when the pointer moves to the next location.

Right. You can find the key-value pair allies comma 28. So, that will be printed. So, here the duplicates are allowed, alright. So, we can check that.



So, we will go to the C++ code. So, here also I have included map, alright. So, maybe what we can do, we can test the previous case once we run this code. So, it is a multi-map. So, multi-map when we run the code.

You can see what is happening. Alright, so 'allies' will be printed 2 times, but the age is 25 and 28. That means duplicates are allowed. So, what we can do in the previous program? Press any key to continue. We will go to the previous program. So, we will just do. We put age map Bob.

Next one. Next line. Line number 9. Just copy line number 8 and paste it. Next line.

Just put enter. Age map. Open the braces. Yes. Same Bob.

Now, what we will do? We will put 35. Right. This I am testing in equal to 35. Semicolon.

Fine. Right. So, what I did? I included the duplicator. So, now let us see when you run the code, execute, compile and run.

So, you can see that, right. So, that was overridden, right. Age map of Bob equal to 30, line number 8, right. So, the duplicate we cannot insert. Maybe we will try one more case of multi-map.

So, here what we will do, it is already an alias, right. So, we know that if I write alias comma 30, that will also be printed. Maybe, if you want, we can do that. Multi-age map insert. Let us use Control C. Enter.

Control V. We put 25, 28, 31. Right. So, can you guess the answer? You know that. Everything will be printed.

Right. That is in the case of a duplicate. Right. Let us run the code. Compile and run.

Right. So you can see that allies 25, 28 and 31. Right. So this is the case of the multi map. case of the multi-map duplicates are allowed right

so this is what we have done this is duplicate keys in fact we tested so one we tested with the two times allies another one we tested three times allies it was working whereas in the case of map we try to put bob right but you can see that whichever the latest you have entered that will be overwritten so the duplicate will be ignored right so that is a case of map so when there are cases right there are practical example Or examples where the multiple values are associated. So, in this case you can use multimap.

Alright. And you can see that the loop was iterating through all key value pairs. Right. The for loop that we had seen. And in this case it will include duplicates also.

So, whereas in the case of a map. So, we know that that will be ignored. Right. So, case of a Bob. Right.

One time we put 25. Another time Bob 30. 25 were ignored. right that is over because of the duplicate. So, this is the difference between map and multi map, right.

So, in the next lecture I will start with introduction to unordered containers, right. So, with this I am concluding this particular lecture. Thank you.