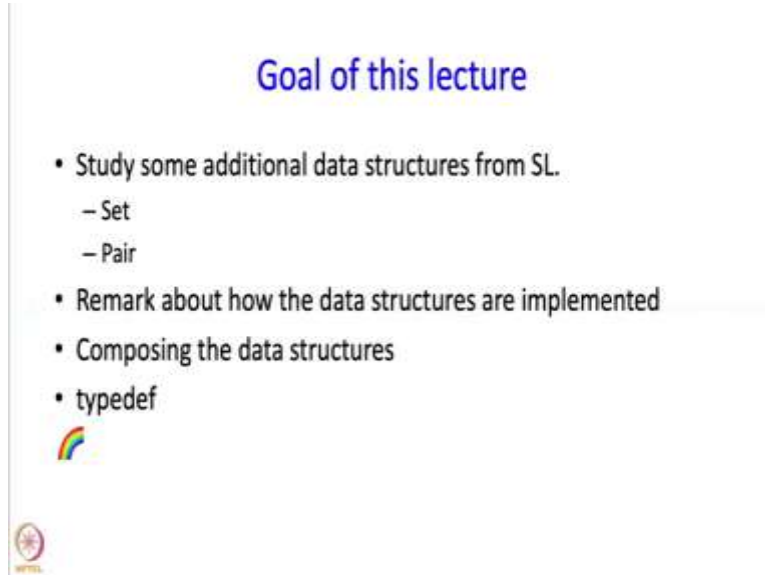




An Introduction to programming Through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Lecture No. 24 Part- 2
Data structure based programming
Set and pair classes

(Refer Slide Time: 00:19)



Goal of this lecture

- Study some additional data structures from SL.
 - Set
 - Pair
- Remark about how the data structures are implemented
- Composing the data structures
- typedef



In the last segment we saw an introduction to this lecture sequence. In this segment, we are going to study the set and the pair classes or data structures from the standard library.

(Refer Slide Time: 00:31)

The set class from the Standard Library

```
#include <set>

int main(){
    set<int> students;
    while(true){
        int rollno; cin >> rollno;
        if(rollno < 0) break;
        students.insert(rollno);
    }
    int s; cin >> s;
    cout <<students.count(s)<<endl;
    for(auto r : students)
        cout << r << endl;
}
```

- Can have set of arbitrary types
- Can insert into set
- count: > 0 if element is present in the set
- Can iterate over set
 - Printing happens in increasing order
 - Lexicographic order for strings
- Many other operations, e.g. remove elements from the set
- See various on-line documentations

Unordered_set:

- Faster
- Some features absent

So, I am going to introduce the set class through an example, through a program. So, first of all, when we write a program we will need the usual things like include simple CPP but if we want to use the set class we should also have `#include<set>`. Then we will have the main program and in the main program I can define a set as shown here. So this is saying that I want a set of integers. I want a variable called `students` which is going to be representing, which is going to be containing a set of integers. And I am not restricted to a set of integers. I might want a set of strings for example. So, here I am thinking of representing the students using their roll numbers. If I wanted to represent them using their names then a set of strings would be the right, the right class for me to create, the right variable for me to create. Now, let me just go ahead with the program, so this program is going to read a bunch of student roll numbers.

So, a loop which goes while true there is no exit at the top, but what we are going to do is, we are going to read in the roll number. So the variable roll number is going to be read from the keyboard and we are going to have this convention that if the roll number that that is read in, is smaller than 0, then that is going to be a signal that all the roll numbers have been typed in. So in that case we are going to break. Otherwise what we typed in is a valid roll number and we just want to insert it into our student set. So that is accomplished by the statement, `students.insert(roll number)`. So into a set what we have seen here is that you can insert something and since this is a set of integers, you can insert an integer. Now we have inserted all the roll numbers into our set

called students. What can you do with this next? So here I am going to read in 1 more roll number. So let me call it S. So I write into S from the keyboard.

Now, I can check whether this roll number is present in my student set. How do I do this? So for this we have this function, where this member function called count. So students that, students.count(S) is going to return the number of times S appears in the set students. So, if S was typed, then this would evaluate to 1. So, if S is not present, then this would appear 0 times. So you really should of count, as taking the value 1 or 0 but yeah so that is it. This is the set and so really no element can appear more than once here.

Then we are going to do something interesting. We are going to print out all the elements of the set and this is really very similar to what we did for maps. So, over there we iterated, so we iterated over this set and now this r is going to be each element of this set. So, we are going over all the elements in the set students and we are going to refer to each element using this variable r. So, I can iterate over sets. So that is one of the operations that I can do in with a set and given that operation what can I do with r? Well each element is a roll number and I can print out that roll number directly.

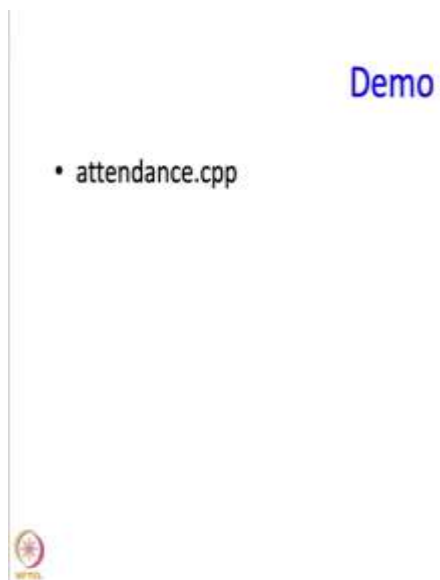
Now, there is some interesting point over here. So this set implicitly is an ordered set. So, when I go over the set elements, when I iterate over the set elements, I am going to encounter the elements in increasing order. So, this less than order should be defined on this data type and indeed less than is defined on this data type and, in fact, I am going to get in increasing order. If I had had a set of strings, then I would have got them in lexicographic order.

So this is this is something nice. So I will now be able to print all the set elements but they will be in increasing order or lexicographic order whatever is consistent with the less than order which must be defined on this data type. So that is it, so that is it for this program and in addition, the set class or the set data structure has many other operations defined. For example, you can remove an element after inserting it into a set and this is done by the member function erase but that is not the only function. There are other functions also available and there are various online documentations. So just do a search on the net and you will get all the other operations which are available. So we are not going to, we are not going to rely on all those operations. I mean this is just a this is just a very simple minded introduction to this class and so

for understanding, for answering, the questions in the exams as well as answering the question in the quiz, whatever we have discussed over here is going to be adequate.

Let me just point that there is something also called unordered set. So in this, there is no order, so if I if I am going to iterate over them, I will get the elements in some order which you cannot really make any sense of. So there are numbers, may be the numbers will come out increasing but may be they will not. They will come in some random order. But this set turns out to be faster. It is a little tricky and I am not going to talk more about it and some of the features that we have talked are mentioned over here like this iteration order is not available if you, if you create the set of, if you create an unordered set.

(Refer Slide Time: 07:11)



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

int main(){
    set<int> students;
    while(true){
        int rollno; cin >> rollno;
        if(rollno < 0) break;
        students.insert(rollno);
    }
    int s; cin >> s; cout << students.count(s) << endl;
    for(auto r : students)
        cout << r << endl;
}
```

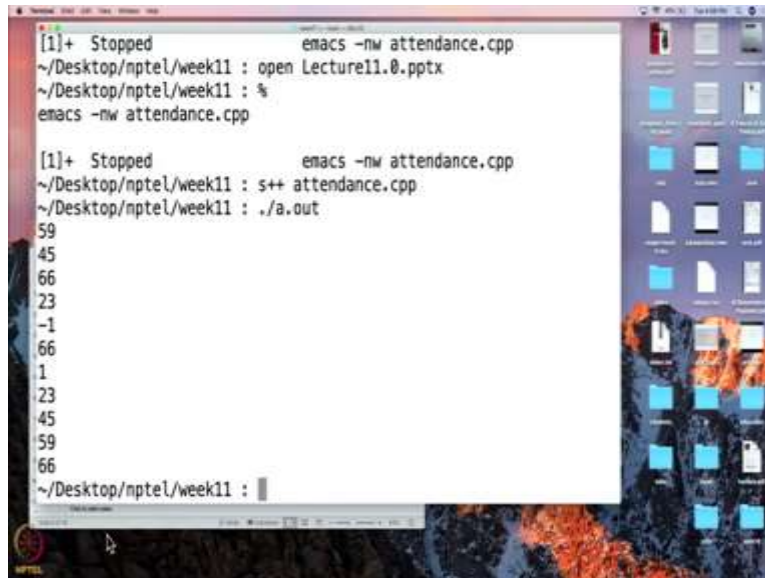
~UU-:—F1 attendance.cpp All L14 (C++/l Abbrev) 4:54PM 1.3

```
24b
100a
~/Desktop/nptel/week11 : %
emacs -nw attendance.cpp

[1]+ Stopped                  emacs -nw attendance.cpp
~/Desktop/nptel/week11 : open Lecture11.0.pptx
~/Desktop/nptel/week11 : %
emacs -nw attendance.cpp

[1]+ Stopped                  emacs -nw attendance.cpp
~/Desktop/nptel/week11 : open Lecture11.0.pptx
~/Desktop/nptel/week11 : %
emacs -nw attendance.cpp

[1]+ Stopped                  emacs -nw attendance.cpp
~/Desktop/nptel/week11 : s++ attendance.cpp
~/Desktop/nptel/week11 : ./a.out
59
```



```
[1]+ Stopped                  emacs -nw attendance.cpp
~/Desktop/nptel/week11 : open Lecture11.0.pptx
~/Desktop/nptel/week11 : %
emacs -nw attendance.cpp

[1]+ Stopped                  emacs -nw attendance.cpp
~/Desktop/nptel/week11 : s++ attendance.cpp
~/Desktop/nptel/week11 : ./a.out
59
45
66
23
-1
66
1
23
45
59
66
~/Desktop/nptel/week11 : 
```

So, I am just going to give you a demonstration of sets. So, let us go to that. So you can see this is the same program which we had seen earlier on the slides. So, let me compile it. So, I have compiled it. Now, I am going to run it and now as I run it, the first thing if you remember I have to insert students into that set and I have to end by printing, by giving in some negative quantity. So, maybe I will put 59 as the first roll number. This is the first student that enters.

So, let me just move this so that it is clearly visible. Then may be 45 enters then may be 66 enters, 23 enters and then may be that is the end and so I will give minus 1. So at this point the program knows that all the students that were supposed to enter have been have entered. Now, what was the student doing? What was the program doing? Well the program is now expecting me to type in a roll number and it is going to tell me whether that roll number is present in that set or not.

So, let us say I typed 66. So, what, let us see what happens? So, when I typed 66, the first thing it type if you remember is whether or not, what is the count of 66 in this in this set? So, in fact, the count of 66 is 1 because 66 is present over here. And after that, it printed all the roll numbers in the set in increasing order. So, this is the increasing order over here 23, 45, 59, 66. We did not type them in that order but it prints them out in increasing order.

(Refer slide Time: 09:13)

Pair

- **General form:** `pair<t1,t2> var;`
 - predefined struct with two elements, of type t1 and t2
- ```
pair<int,string> p1,p2;
p1 = pair<int,string>(50,"fifty");
p2.first = 60;
p2.second = "sixty";
cout << (p1 < p2) << endl;
```
- lexicographic sorting order



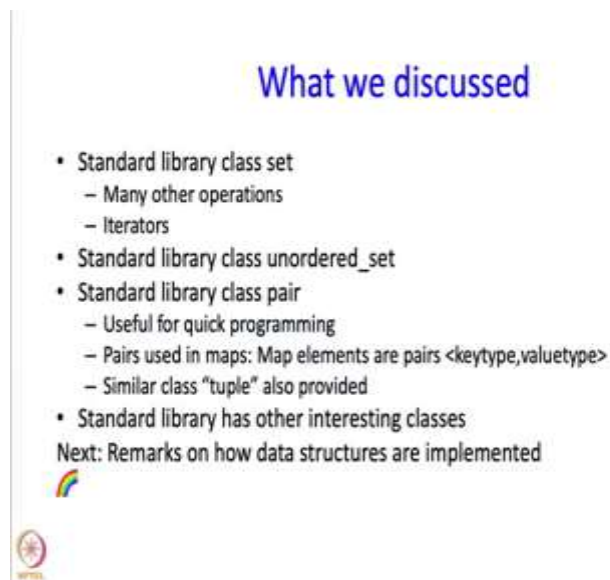
Alright, so let us go back to the slides and let us move on to pairs. The general form of how you declare a pair is this. So you say pair, that you name a type and another type and then you name the variable, give the name of the variable that you are trying to create. So, what does this do, well it creates a struct and it, the struct only has 2 elements. The first element which is actually called the the data member is called first, has to have type t1 and the second has to have type t2.

So, you can think of this as being a struct which is pre-defined for you. If you just you do not have to actually say struct and then give members and all of that. This struct is sort of pre-defined because of this statement over here. So, here is an example, I might say `pair<int, string> p1, p2;` So, this creates 2 variables p1 and p2 of type pair of int string. So p1 has an int part and the string part and p2 also has an int part and the string part and, in fact, to get to the int part of either I have to use the member first or I so I have to say `p1.first` or `p1.second` to get to the string part.

Now, there is a nice feature. I have this constructor also available. So again this is all pre defined. So here I am I have to type the initial value of the integer that I want to create, the initial value of the string that I want to create. So what would this do, this would create a struct p1 whose first part is 50 and second part is 50 but in words. Alternately, I can do this as well. I can initialize `p2.first=60` and I can initialize `p2.second=60` as well. So the string part would be set to this 60 and finally there is an a very interesting convenience provided. I can compare pairs.

So, I can write something like `p1 < p2`. So, what does this do, this is going to be less than if the first part of `p1` is less than the first part of `p2`. If they are both equal, then it is going to go to the second part. So, what do you think is going to happen in this case? So, over here the first part of `p1` is 50 and the first part of `p2` is 60. So, this comparison is going to come out true. So, this will cause a 1 to be printed or to be printed. So, yeah, so the comparison happens according to the lexicographic sorting order.

(Refer Slide Time: 12:09)



The slide is titled "What we discussed" in blue text. It contains a bulleted list of standard library classes and their uses. The list is as follows:

- Standard library class set
  - Many other operations
  - Iterators
- Standard library class `unordered_set`
- Standard library class `pair`
  - Useful for quick programming
  - Pairs used in maps: Map elements are pairs `<keytype, valuetype>`
  - Similar class "tuple" also provided
- Standard library has other interesting classes

Next: Remarks on how data structures are implemented

At the bottom left of the slide, there is a small rainbow icon and a circular logo with a star inside.

Alright, so what have we discussed in this segment. So, we have discussed the standard library class set and in addition to the operations that I showed you, there are many other operations available. It also allows iterators and we also discussed or rather we just mentioned the standard library class unordered set. Then we discussed the standard library class pair. Now, pair is something that has meant to be used for quick programming, rather than having to take the trouble of creating a struct.

So, defining a struct then using it, you can just sort of create the struct on the fly. Yeah, so that is what pair is going to be for. And by the way pairs are used in maps. So, if you remember a map itself has 2 parts, a keytype, a key part and the value part. So actually maps are indeed sequences, ordered sequences of key type value type. So order according to key type and, in fact, you have seen this when we printed the contents of maps in the last lecture.



Remember lecture in which we had a map from names of countries to their populations, well so there we got things we wrote things country name, we wrote things like variable.first and that first was the country name and variable.second and that second was the population. So pairs are actually used and are parts of maps. C++ also allows you to define a similar class called tuple, oh by the way here is one quick use that you might put pairs to. If you want to return 2 objects, then you can just return it as a pair and you can receive it as a pair in your calling program as well. So rather than having to define a proper struct this is sort of a quick way doing things and, of course, the fact that pairs are lexicographically ordered is also very helpful. The standard library also has other interesting classes, something called priority queue is quite interesting but we are not going to look at those in in this in this course. In the next segment we are going to talk a little bit about how these data structures are going to be implemented, but before that let us take a break.