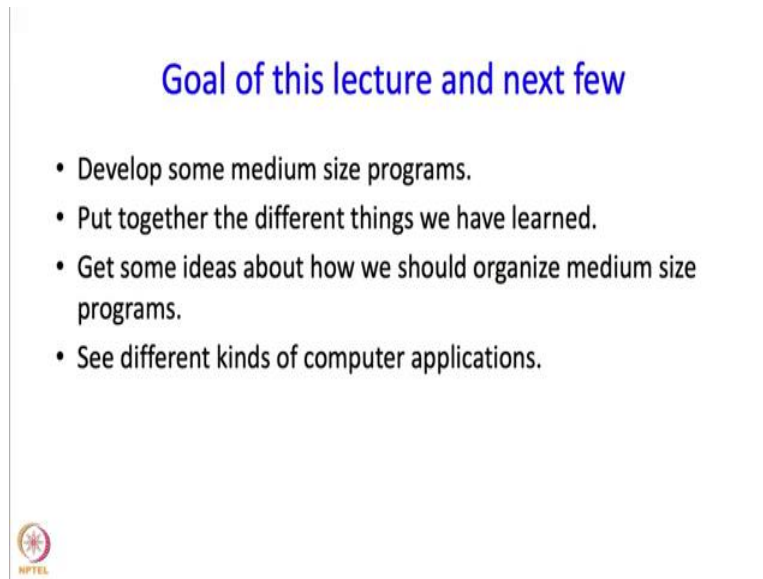


**An Introduction to Programming through C++**  
**Professor Abhiram G Ranade**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology Bombay**  
**Lecture No. 25: Part - 1**  
**Medium size programs**  
**The new marks display program**


Hello, and welcome to the NPTEL course on an introduction to programming through C++. I am Abhiram Ranade and in this lecture, I am going to talk about medium-sized programs.

(Refer Slide Time: 0:39)



Goal of this lecture and next few

- Develop some medium size programs.
- Put together the different things we have learned.
- Get some ideas about how we should organize medium size programs.
- See different kinds of computer applications.



So here is the goal of this lecture and the next. So, we are going to develop some medium-sized programs. So, this will require us to put together the different things that we have learned and in the process we will also get some ideas about how, medium-sized codes should be organized. And along with that, we will also see different kinds of computer applications.

(Refer Slide Time: 1:03)

## Program 1: Enhanced marks display

The program must read data from two files

- **RNFILE:** contains lines, each having:
  - Roll number
  - Name of the student with the given roll number (assume no spaces)
- **RSMFILE:** contains lines, each having:
  - Roll number
  - Subject code (assume does not contain spaces)
  - Marks obtained by the student with the given roll number in the given subject

After this the program must respond to commands from the keyboard



The first medium-size program that we are going to talk about will build upon the marks display sequence of programs we have talked about. So this is going to be an enhanced version of it. So, in this case the program is expected to read data from two files. The first file which I am going to call RNFile, contains lines, each having the following: So, first on each line there is the Roll Number, and then there is the Name of the Student with the given roll number. And for simplicity, we are going to assume that the name does not contain any spaces, but in real life we would have to handle names with spaces as well. But, we are not going to bring that in right now. Then, there is another file called the RSMFile, and this contains lines and each of these lines will begin with a Roll Number, then there will be a Subject Code and again in that subject code will not contain spaces, that is our assumption. And following that, there will be a number which will denote the marks obtained by the student whose Roll Number is given on that line in the subject, whose code is given on the line. Okay so this is the data that our program will read. You will note that I have chosen the names RNFile and RSMFile with some purpose. So RNFile is called RNFile because it has two fields: Roll Number and Name and so the first letters of these are R and N. So we call all that data as residing in the RNFile and similarly, Roll Number, Subject, Marks whose first letters are R, S and M. So, that data will constitute the RSNFile. Now, our program must read these two files, and load up the data that's in these files and after that the program is expected to respond to commands from the keyboard.

(Refer Slide Time: 3:19)

## Commands

Execute commands of the following type until a command to shut down the program is given.

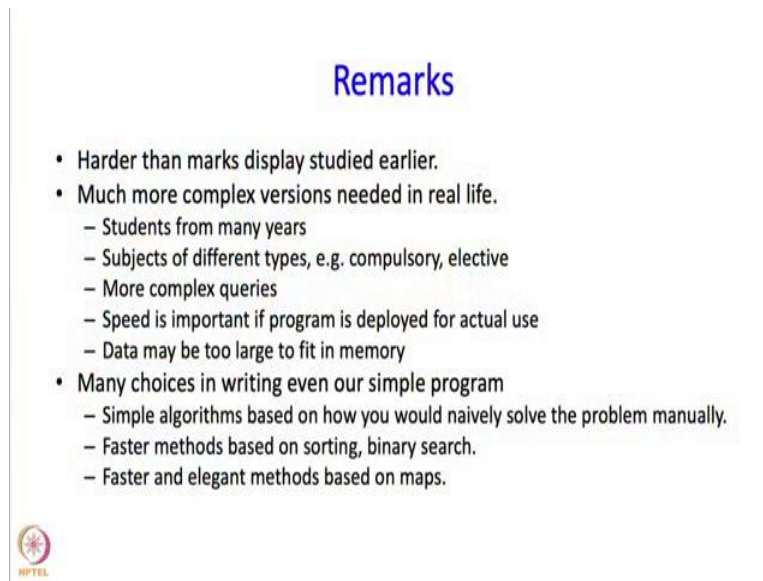
- Print marks obtained by a given student in a given subject.
- Print the rank of a given student
  - Rank determined according to average marks over all courses taken



Okay, so, what are these commands: Well, the commands are the following until and the commands have to be executed until the user types a command to shut down the program. Okay so, until the user types the command which requires shut down, the program will have to execute all the other commands that the user types. So, one command could be that look, tell me the marks obtained by this given student in this given subject.


Okay, so, the student may specify the roll number or maybe the student will specify the name but we want the marks of the given subject to be printed out. Then, we might want the rank of a given student to be printed. Okay, now what is the rank? Well, for the rank, we are going to calculate average marks obtained by the student in all the subjects that the student has taken. So, the student who has the highest average marks will have a rank 1, and the student who has the second highest average marks will have rank 2 and so on.

(Refer Slide Time: 4:29)



**Remarks**

- Harder than marks display studied earlier.
- Much more complex versions needed in real life.
  - Students from many years
  - Subjects of different types, e.g. compulsory, elective
  - More complex queries
  - Speed is important if program is deployed for actual use
  - Data may be too large to fit in memory
- Many choices in writing even our simple program
  - Simple algorithms based on how you would naively solve the problem manually.
  - Faster methods based on sorting, binary search.
  - Faster and elegant methods based on maps.



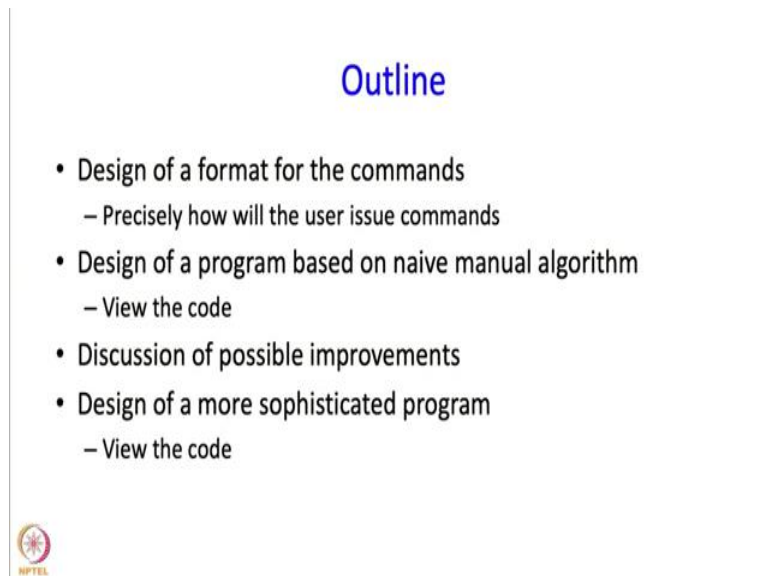
So, let me make a few remarks, so first of all it is clear that this is harder than the marks display programs we have studied earlier. However, by no means this is the hardest possible version. If you look at what might be needed in real life, things are a lot more complicated. Okay, so for example, you may have students from many years, okay, so they may have two students from different years may have the same name or for that matter two students but the same year might have the same name. And, of course, we already said, that student's names may have spaces inside them. Then there could be subjects of different types, so for example there could be compulsory subjects, there could be elective subjects. And we may ask, give me the rank in the compulsory subjects or give me a rank and the elective subjects okay. And yeah so there could be more complex queries and if we are doing this in a program that is a deployed, then how fast do we respond is going to be important. So, if this is a part of an information system so a student comes up or a teacher comes up or whoever comes up to your terminal and types in the command, that person expects to get the response reasonably fast if at all possible. And therefore, speed is going to be important, if the program is deployed for actual use. So we are not going to be looking at all of these issues. But we will maybe touch upon some little bit. And this is a really important issue which we will definitely not touch upon. The data may be too large to fit in the main memory of your computer. So, if you have a huge university with lots of students, over lots of years, lots of subjects then the data can be large. And the data may not fit in the memory. And then you would have to worry about, if the command requires data from a certain portion of the disk, how you get it. Okay but, these these are issues that we are not going to deal with.

Before we go on to our program, and our program is simple as compared to what might be needed in real life, though it is complicated more complicated than what we have seen so far. So I just want to point out that, there are many choices in how we might implement our program that we have set ourselves up for. So, there could be a very simple algorithm and this might be based on how you would naively solve the problem manually.

Okay so, we will try and mimic what you might do if you had these files say as notebooks and somebody came to you and asked the questions. But, of course, even here there is an issue, if you are clever then you may solve the problem in one way and if you are not so clever you may solve the problem in another way. So we will look at some simple ideas first.


Then, once we understand what exactly we need to do, we can try to be clever. And then, we will think about and we will talk a little bit about how we can speed up the program using say, sorting and binary search, which we have already studied. And we have already seen that binary search can be much faster than going through data in the linear fashion. And then, we will talk about faster and elegant methods which are based on maps.

(Refer Slide Time: 8:06)



**Outline**

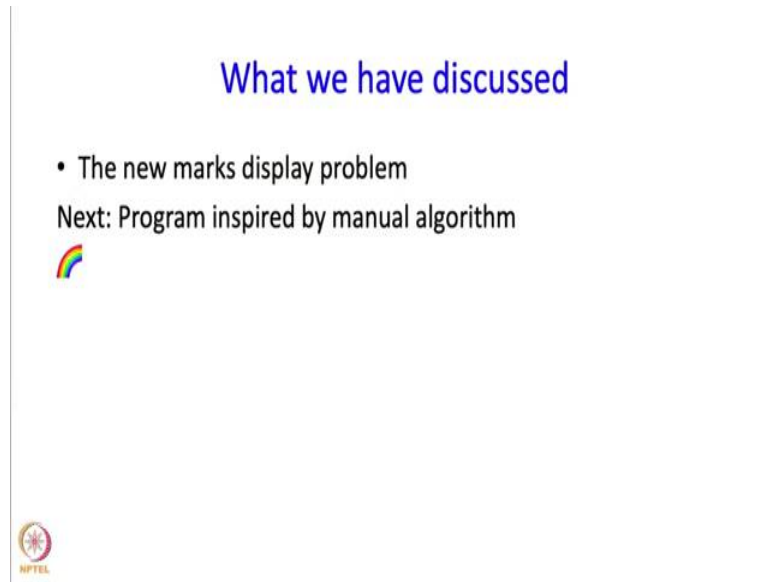
- Design of a format for the commands
  - Precisely how will the user issue commands
- Design of a program based on naive manual algorithm
  - View the code
- Discussion of possible improvements
- Design of a more sophisticated program
  - View the code



So, here is an outline of this lecture, so we will begin by designing a format for the commands. We have discussed what kinds of commands we might need. But now, we will specifically design what exactly is the user is required to type in order to issue a certain command. So, after that, we will look at a program based on somewhat naive manual

algorithm. We will look at its code and then we will discuss possible improvements. And then we will design more sophisticated programs, will also look at code.

(Refer Slide Time: 8:52)



Okay, so that brings us to the end of this segment. So, let me just review what we have discussed, so we have discussed the “New Mark's Display Problem” and next we are going to talk about the program inspired by a manual algorithm. But before that we will take a quick break.