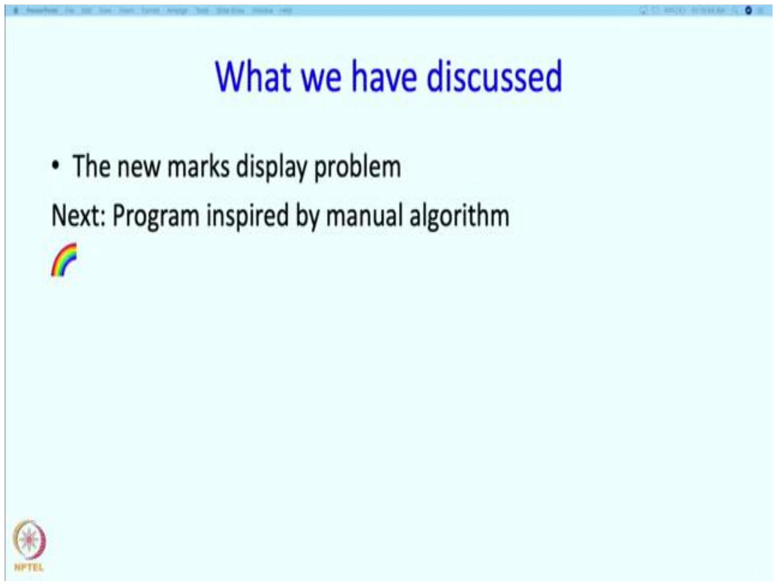
An Introduction to Programming through C++

Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Lecture No. 25 Part – 2
Medium Size Prgrams
Manual Algorithm for new marks display

Welcome back.

(Refer Slide Time: 0:21)



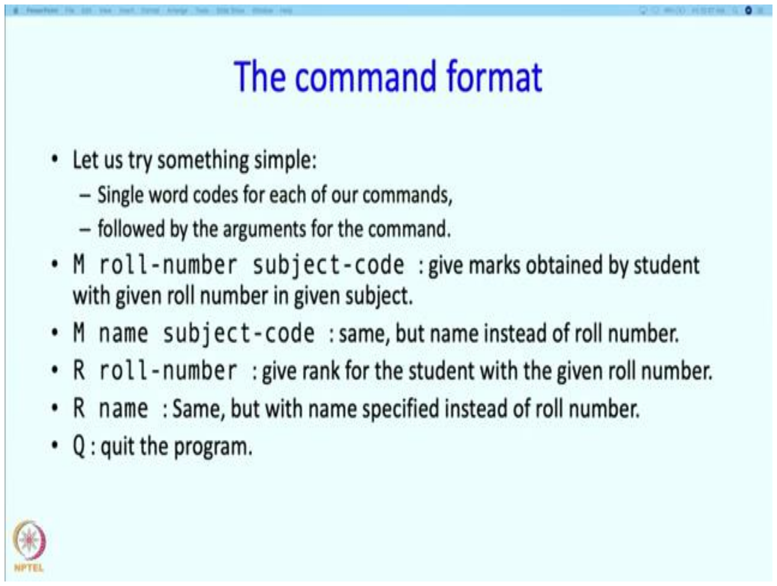
What we have discussed

- The new marks display problem

Next: Program inspired by manual algorithm

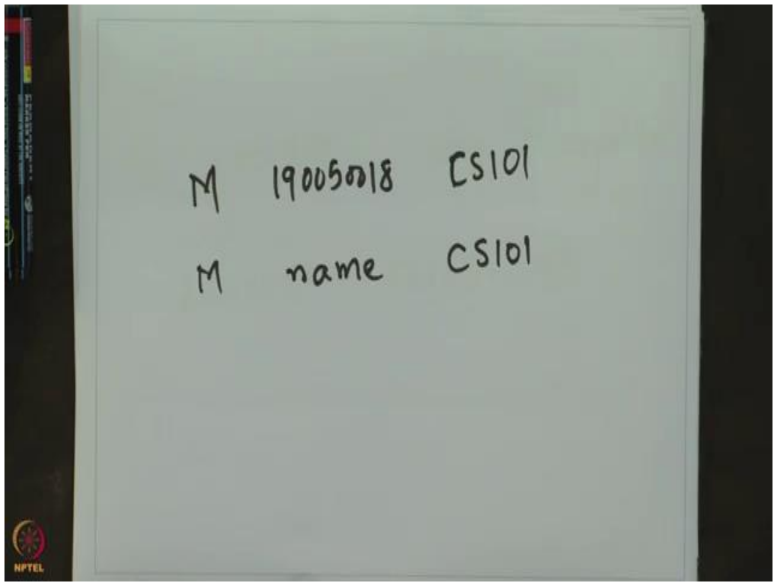
In the last segment we defined the new Marks display problem. In this segment, we are going to come up with a program inspired by manual algorithms.

(Refer Slide Time: 0:32)



The command format

- Let us try something simple:
  - Single word codes for each of our commands,
  - followed by the arguments for the command.
- `M roll-number subject-code` : give marks obtained by student with given roll number in given subject.
- `M name subject-code` : same, but name instead of roll number.
- `R roll-number` : give rank for the student with the given roll number.
- `R name` : Same, but with name specified instead of roll number.
- Q : quit the program.



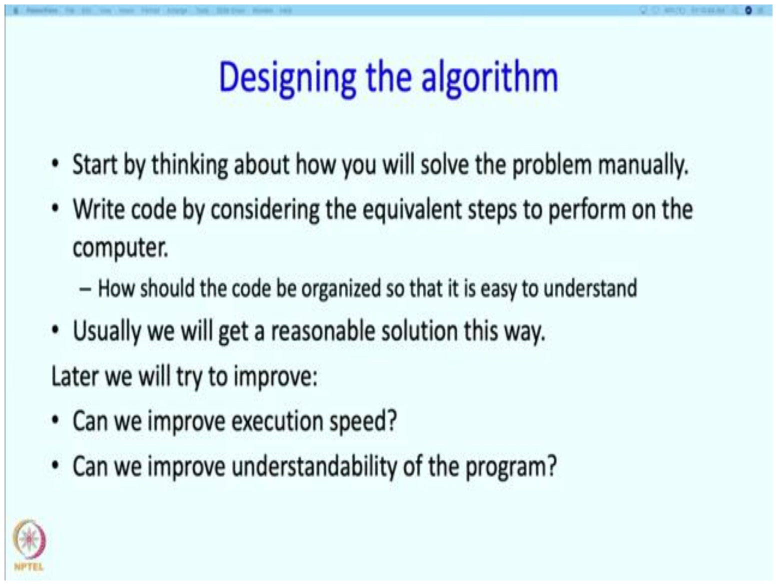
M 19005018 CS101

M name CS101

So, first let us define how the user will communicate with our program. How will the user issue commands. So, we can try something simple. So, let us say we will use single word codes for each of our commands. And then the codes are followed by the arguments for the command. So, for example, we might have a command which looks like M roll number subject code and what this means is – Tell me the marks obtained by the student with the given roll number in the given subject or instead of the roll number the user may type in the name.

So, for example, the user may type in something like this so the user may type in M followed by roll number followed by subject code or M followed by name followed by the subject code. And likewise, we might have a command R followed by roll number. It says give me the rank for the student with the given roll number or R followed by name which says give me the rank for the student with the given name. And you could have a command just a single letter Q, which says quit or exit the program.

(Refer Slide Time: 2:04)



Designing the algorithm

- Start by thinking about how you will solve the problem manually.
- Write code by considering the equivalent steps to perform on the computer.
  - How should the code be organized so that it is easy to understand
- Usually we will get a reasonable solution this way.

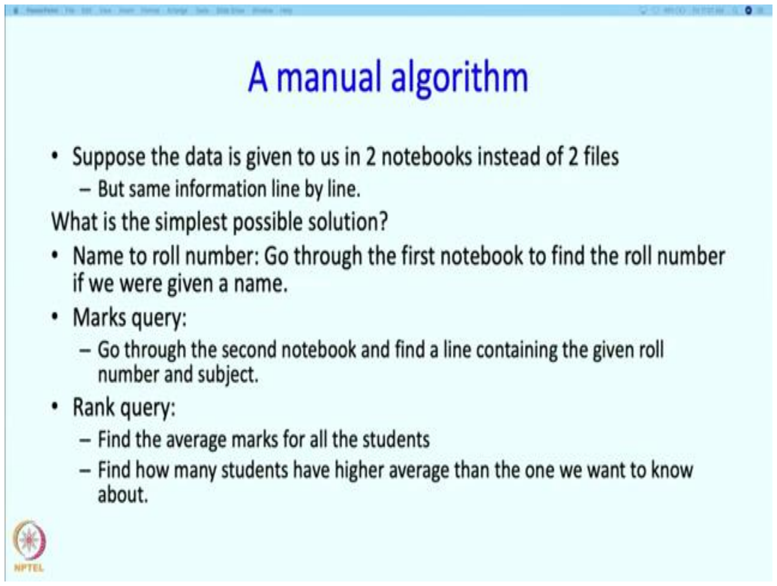
Later we will try to improve:

- Can we improve execution speed?
- Can we improve understandability of the program?

So, now, let us talk about how you might design the algorithm to respond to such queries. And as we have said several times in the course, let us start by thinking about how we might solve this problem manually. So, after we do that, we should consider what is the equivalent of the manual steps on a computer. And that will allow us to write the code. While doing this, we will also continuously keep in mind that we want the program to be easy to understand. And this is something that becomes quite important when you write large or even medium sized programs like the one we are going to write.

Usually this strategy of designing a program by thinking about how you solve the problem manually gives us a reasonable solution but we will try to improve upon what we get as well. So, specifically will ask well can we improve the execution speed. And we would also keep in mind that we also want the understandability of the program to be good. And so, we will think about that as well.

(Refer Slide Time: 3:26)

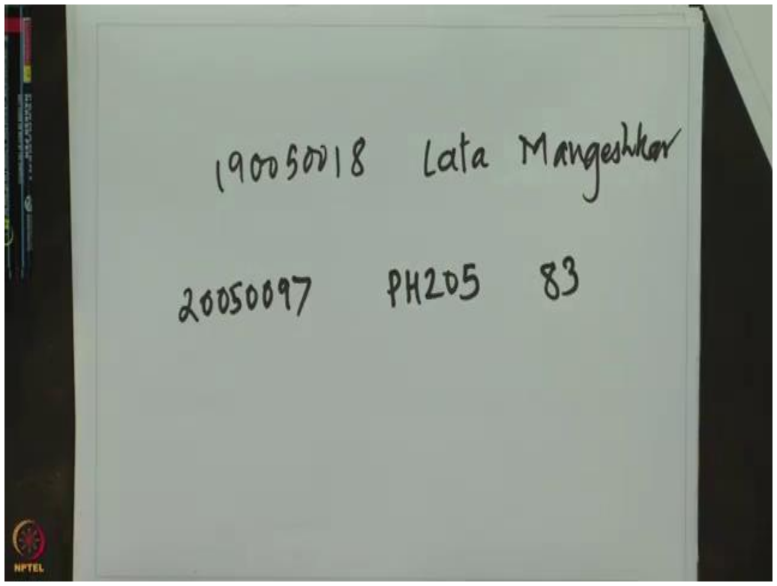


A manual algorithm

- Suppose the data is given to us in 2 notebooks instead of 2 files
  - But same information line by line.

What is the simplest possible solution?

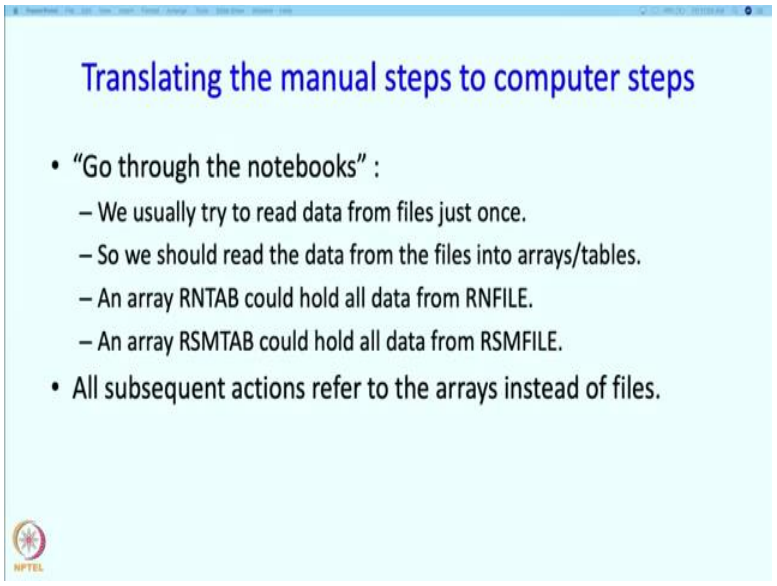
- Name to roll number: Go through the first notebook to find the roll number if we were given a name.
- Marks query:
  - Go through the second notebook and find a line containing the given roll number and subject.
- Rank query:
  - Find the average marks for all the students
  - Find how many students have higher average than the one we want to know about.



So, let us think about solving the problem manually first. So, suppose the data is given to us in two notebooks instead of two files. And what appears in the notebook is saying the first notebook is lines of this form, so roll number followed by name. In the second file we are expecting to see roll number followed by subject code followed by Marks. So it will contain lines of this far. So these were the lines which are appearing in files but instead of this let us say they appear in notebooks so that we can think of a manual algorithm.

So, what is the simplest possible solution? Well suppose we want to translate from the name to the roll number. We should go through the first notebook to find the roll number. If you have given the name. So, that is simple enough. And then if we are given the marks query, that is M followed by roll number or name followed by the subject code. Well you have to go through the second notebook, which contains lines of this form and we should look for the roll number and the subject being present in the same line. And then we should just report the marks. The rank query, well, we have to find the average of the average marks for each student. What I mean by this is for every student we want to find an average over all the subjects in which the student has marks. So, if a student has say 90 marks in physics, and 80 marks in mathematics, then there will be two lines in the second notebook corresponding to that student. So, we have to find those two lines, and we have to take the average which in this case is going to be 90 plus 80 or 85. So that is going to be the first step. For every student we must look at the subjects he or she has taken, and find the average marks obtained by that student. So, this is going to be a somewhat involved task. Then we will have a table so to say, of the average marks obtained by every student. And then in this table we have to figure out how many students have higher average than the one whose rank we want to know. So, that will tell us exactly the rank.

(Refer Slide Time: 6:22)



Translating the manual steps to computer steps

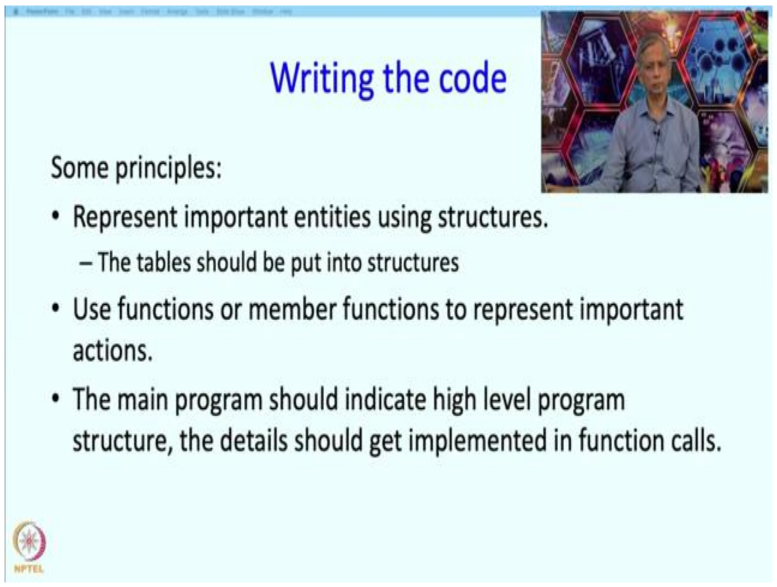
- "Go through the notebooks" :
  - We usually try to read data from files just once.
  - So we should read the data from the files into arrays/tables.
  - An array RNTAB could hold all data from RNFILE.
  - An array RSMTAB could hold all data from RSMFILE.
- All subsequent actions refer to the arrays instead of files.

So those were the manual steps. How do we translate the manual steps into computer steps? So, what does it mean to say go through the notebooks? Well, nominally it would mean read through the file, but usually on a computer, our idea is, we read the file just once. We do not read it repeatedly because reading files is often too slow. So, what we do is we read the files and then we put the data into some array or something like that in our computer.

So, we will read the data from the file into an array or informally, we can think of table associated with the array and say we designate an array RNTAB to hold all the data from the file RNFILE. If you remember RNFILE file was holding roll number and name data. So we will have an array RNTAB which will have, which will consist, which will be an array of structures. And the first member in the structure will hold the roll number. And the second member will hold the name.

Then we also have an RSM file, which contains lines of the form roll number subject marks. So, we will designate an array let us call it RSMTAB. TAB for table R S M for Roll Number subject marks. And this will be an array of structures, each structure containing three data members - roll number, subject and marks. And the subsequent actions that we perform, refer to arrays rather than the files.

(Refer Slide Time: 8:17)



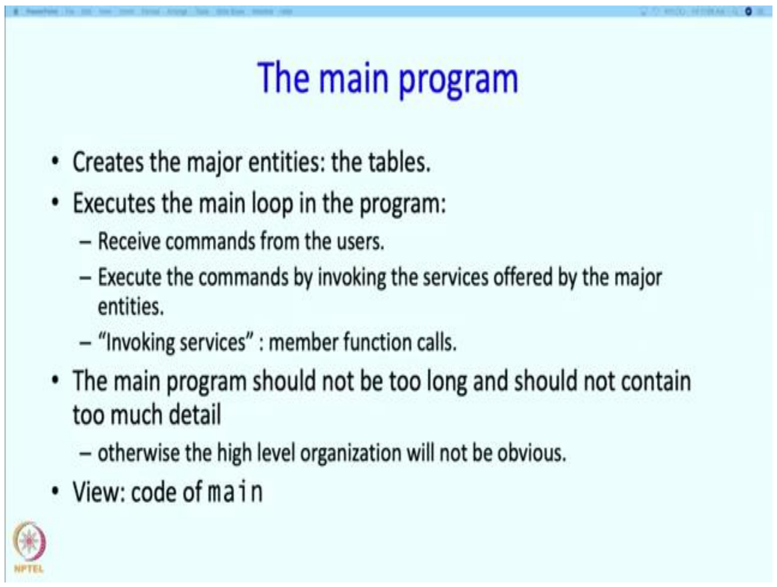
Writing the code

Some principles:

- Represent important entities using structures.
  - The tables should be put into structures
- Use functions or member functions to represent important actions.
- The main program should indicate high level program structure, the details should get implemented in function calls.

So, let us think about writing the code, since, we have a fairly clear understanding of what we are going to do manually. So, now when we write the code, we have to keep some broad principles in mind. And we have talked about these principles earlier. So basically, we would like to represent important entities using structures. So, the tables that we define should really be put into structures. So I mean, something slightly different than what I said earlier. What I said earlier was that each entry in the table or each element of the array of the array will be a structure. Now, I am saying that the entire array should in addition be put in a bigger struct. So, that is sort of the general idea that we are going to put important entities inside a struct, because the struct kind of encapsulates that entity, and we can put member functions now more easily. So, we can associate member functions with that with that entire entity. So, RNTAB and RSNTAB are going to be important entities. So, although they contain inside them other structures, we will put the whole thing also inside a structure of an appropriate type. So, we will see this. And we will use functions or member functions to represent important actions. And the main program should indicate high level program structure. So, what I mean by that is the main program should contain function calls which, say, cause a query a Marks query to be processed or another function call to process say the rank query, but the main program should not should not contain details. So, the details should be implemented in function calls or member function calls associated with the structures which are holding our important entities.

(Refer Slide Time: 10:56)



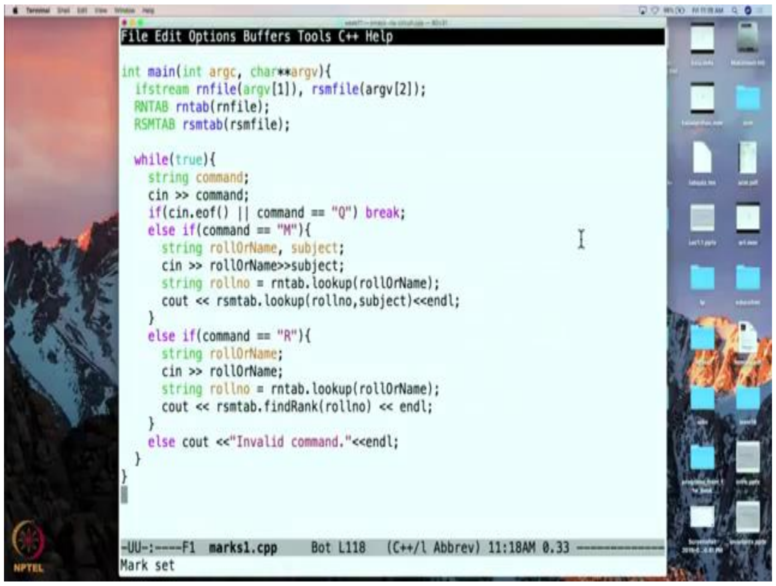
The main program

- Creates the major entities: the tables.
- Executes the main loop in the program:
  - Receive commands from the users.
  - Execute the commands by invoking the services offered by the major entities.
  - "Invoking services" : member function calls.
- The main program should not be too long and should not contain too much detail
  - otherwise the high level organization will not be obvious.
- View: code of `main`

So, let us look at the main program first. So, the main program should create the major entities, in our case the major entities are the table, so these should get created by the main program. Then the main program should execute. The main loop and the main loop should receive commands from the users. And the main loop should execute the commands by invoking the services offered by the major entities. So, when I say invoking services, I mean just a function call. The main program should not contain details, the details should be in the member function or the function calls.

The main program should not be too long and should not contain too much detail. So, I said that the main program should create the tables, but the creation should really happen in the constructor associated with the entities. So, the details of how the tables, or how the major entities are constructed should also be hidden, and the main program should really contain the high level aspect. Why is this? Well, if we put the details in the main program then the high level organization will not become obvious. We do want to write down the details, of course. But we also want to indicate what the high level organization is. And therefore, we should have the main program sort of spell out the higher level organization and from the main program, if you want to know the details about something, then you can you just have to follow up that function call or member function call and the details and then the body of the member function call or the function can we tell you how exactly that function is going to be implemented. So, next we are going to view the code of mean to see how the main program is going to be written to follow these principles.

(Refer Slide Time: 13:18)



```
int main(int argc, char**argv){
  ifstream rnfile(argv[1]), rsmfile(argv[2]);
  RNTAB rntab(rnfile);
  RSMTAB rsmtab(rsmfile);

  while(true){
    string command;
    cin >> command;
    if(cin.eof() || command == "Q") break;
    else if(command == "M"){
      string rollOrName, subject;
      cin >> rollOrName>>subject;
      string rollno = rntab.lookup(rollOrName);
      cout << rsmtab.lookup(rollno,subject)<<endl;
    }
    else if(command == "R"){
      string rollOrName;
      cin >> rollOrName;
      string rollno = rntab.lookup(rollOrName);
      cout << rsmtab.findRank(rollno) << endl;
    }
    else cout <<"Invalid command."<<endl;
  }
}
```

```
-UU-:----F1  marks1.cpp     Bot L118    (C++/l Abbrev) 11:18AM 0.33 ------------
Mark set
```

So, here is our marks, our marks program, Mark display program and here is our main program. So, the way we are going to organize the main program is that it is going to receive the names of the files as command line arguments. So, the first, so the zeroth command line argument will be simply ./a.out. Then the first command line argument following that will be the name of the rnfile. So, argv is the first command line argument, and then the second command line argument argv2. So, argv1 when is the first command line argument argv2 is the second command line argument. And the second command line argument will give the name of the rsmfile. Now, as we have said earlier, to read the files themselves, we have two associate streams in the files. So, we are going to do that. And this is how it is done.

So we are going to define ifstreams, we are going to define int ifstream called rnfile. And this rnfile is going to be an ifstream and the actual file will be in this argv1, what the user types in when the program is invoked as the first argument, the one after the zeroth argument and rsmfile is going to be the stream which will be used inside the program. And this rsmfile will get data from argv2 which is the second name, the second file name which is typed in by the user. So, the user will type the ./a.out followed by the first finally and then the second file name. Then we said so, then we said that the main program should create the major entities. So, the major entities are

RNTAB and RSNTAB. And RNTAB, if you remember, is the table which contains information about roll numbers and associated names. And for that, we need rnfile. So, the way we initialize this is by calling the constructor for RNTAB. And we pass the file rnfile to it. So the details of this we will see soon. But notice that over here we are simply calling the constructor and we are supplying to the constructor whatever information is needed in order for that entity to be constructed. So, the information needed is the stream associated with that file. So, inside that RNTAB constructor the data will be read from the stream and the table will be created.

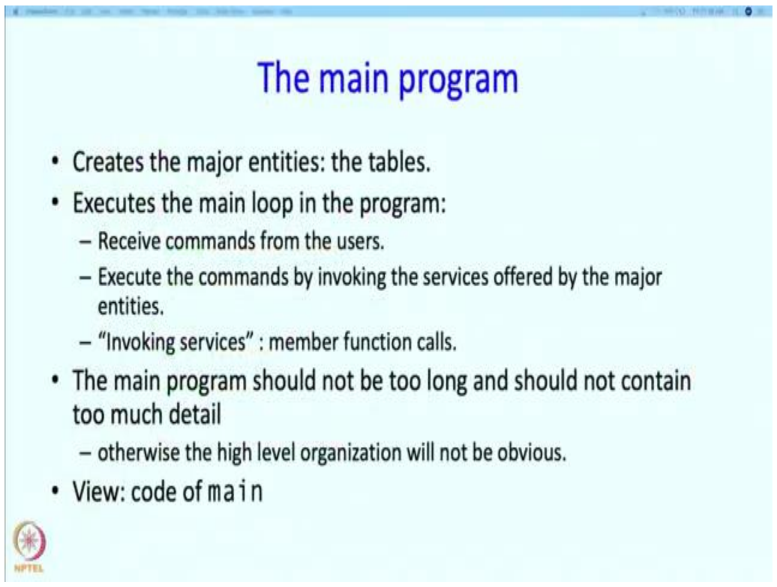
Similarly, the second important entity is this RSMTAB, the roll number subject marks tab and for this we need the RSMFILE. So, again we are calling the constructor. And to that constructor we are passing the stream which is associated with the actual file that the user has typed in on the command line. So, that was the first thing we said that the main program should do. And then the main program should have a loop. So, let us see what this loop is doing. So the loop runs indefinitely until an appropriate command is typed. So, let us see that. So, we have a string variable command and we will read into that variable. So what do we read? Well, we are going to read first the command code or other command codes where m, q and r. But if you remember, the user can also type in control D to indicate end of the standard input. And so what this is saying is that after this reading have you found that cin has ended or was the command Q. So, if the command was Q or if the cin had ended, then we should just break out and that is how the loop is going to be terminated and we come out of the loop and then the program itself terminated. Otherwise we check if the command is an M. So, if the command is an M, then we know that it is a Marks query. So, if it is a Marks query then we are supposed to get in a roll number and a subject name. So, this is how we get that.

Now here, we are simply reading in whatever the user types into the variables roll or name and subject So, if you remember, our Marks command could be marks followed by name or the roll number followed by subject. So, we are just reading in one word into name and an expert into the subject and here we are making a compromise. We are simplifying our problem a little bit. So, we are saying here that name, if the user types a name it consists of a single word. Otherwise what we will need to do is we need to do something else. We will need to read in the entire line. And we will need to extract the name and a subject from it. So, this is something that can be easily done. But as it is, our program is reasonably complicated as we will see. And therefore, we are going to skip this business about reading in names which contain spaces. So, I will leave it as an exercise for you to modify this and I will also present the solution for it. So you will eventually get the code for doing this as well. But just to keep today's discussion simple, we will assume that the name or the roll number that is given to us is just one word. So, we get the name, so we get the roll number or name into a single string. We get the subject. So this is where we do it. So, from that from the keyboard and then we will use of our table RNTAB and we will look up in it to get the roll number if this was a name.

So, we are going to send whatever it is the roll, roll number or the name to our roll number name table and then we will get a roll number for it. So, this roll or name could either be a roll number or name in which case. So, if it is a roll number then lookup has to do nothing, it just returns it back if it is a name then this lookup will have to return back the corresponding roll number. And then, we are going to invoke the lookup function this time in RSMTAB which will give us the student with the specified roll number in the specified subject and that we are directly going to print out.

So, that is how this Marks query will get processed. If the command was a rank query, then the processing is very similar except that there is no subject code needed. So, we are just going to get the roll number or name following the R and again, we are making this compromise assumption that it is going to be a single word. And then we are going to pass that roll number or name and get the roll number. As we said if it if already roll number then we will stay with it. And then we will find the rank for of this given roll number and then we will print it out. If the first word typed in by the user is neither Q nor M nor R, then we are going to print out a message saying that you typed in an invalid command. And we will continue the loop but will give the indication, given an indication to the user saying that look you printed out you typed in it invalid command.

(Refer Slide Time: 22:34)

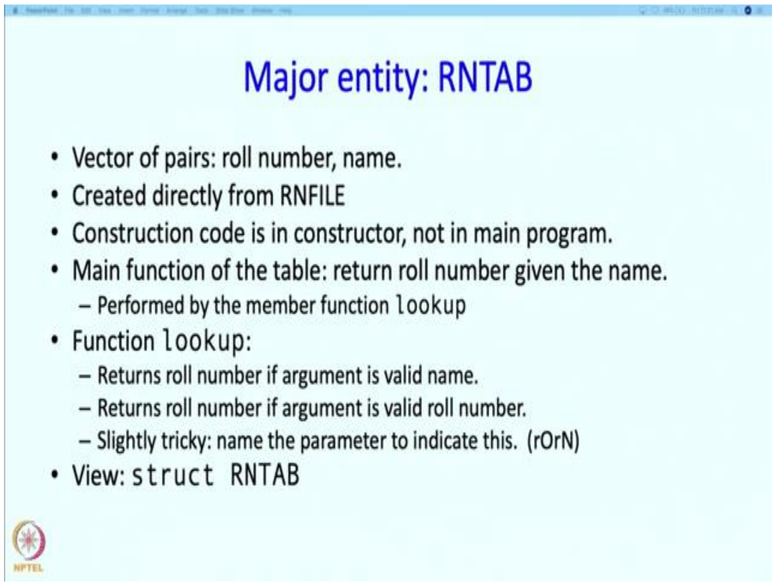


The main program

- Creates the major entities: the tables.
- Executes the main loop in the program:
  - Receive commands from the users.
  - Execute the commands by invoking the services offered by the major entities.
  - "Invoking services" : member function calls.
- The main program should not be too long and should not contain too much detail
  - otherwise the high level organization will not be obvious.
- View: code of `main`

So, let us get back to our presentation. So, we have looked at the code of main. Next comes the major entity that we created - RNTAB.

(Refer Slide Time: 22:45)

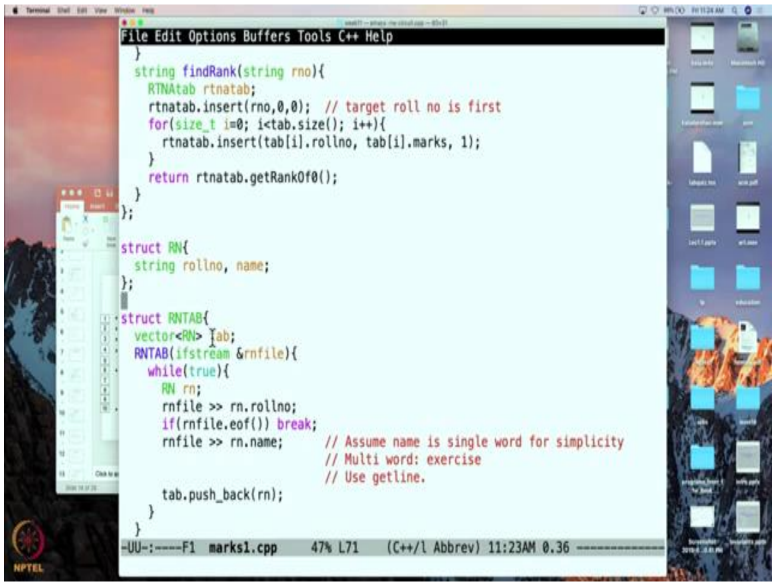


Major entity: RNTAB

- Vector of pairs: roll number, name.
- Created directly from RNFILE
- Construction code is in constructor, not in main program.
- Main function of the table: return roll number given the name.
  - Performed by the member function `lookup`
- Function `lookup`:
  - Returns roll number if argument is valid name.
  - Returns roll number if argument is valid roll number.
  - Slightly tricky: name the parameter to indicate this. (rOrN)
- View: `struct RNTAB`

So, as we say it contains vectors of pairs: roll number and name. So, each element is itself going to be a struct. And we are going to create this directly from rnfile. And the construction code is in the constructor, not in the main program. The main function of this table is to return the roll number given the name. And this is performed by the member function lookup. So, what does function lookup do? Well, it returns the roll number if the argument is a valid name. So for this, we are just going to look through all the names which are stored in this vector, and its return and it is going to return a roll number if the argument is a valid roll number. This is slightly tricky in the sense that the name could either be a roll number or a name and so we are going to have a parameter R or N to indicate this, this this possibility. So, let us now look at the code for this entity the struct RNTAB.

(Refer Slide Time: 24:05)



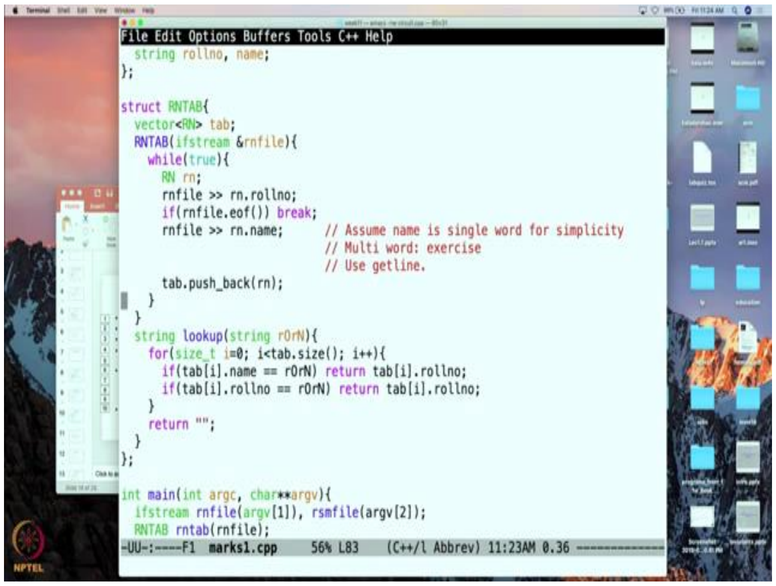
```
File Edit Options Buffers Tools C++ Help
  }
  string findRank(string rno){
    RTNAtab rtnatab;
    rtnatab.insert(rno,0,0);  // target roll no is first
    for(size_t i=0; i<tab.size(); i++){
      rtnatab.insert(tab[i].rollno, tab[i].marks, 1);
    }
    return rtnatab.getRankOf0();
  }
};

struct RN{
  string rollno, name;
};

struct RNTAB{
  vector<RN> tab;
  RNTAB(ifstream &rnfile){
    while(true){
      RN rn;
      rnfile >> rn.rollno;
      if(rnfile.eof()) break;
      rnfile >> rn.name;      // Assume name is single word for simplicity
                              // Multi word: exercise
                              // Use getline.
      tab.push_back(rn);
    }
  }
-UU-:----F1  marks1.cpp     47% L71    (C++/l Abbrev) 11:23AM 0.36 ------------
```

So this is our structure RNTAB. So, what does it contain? So, it is a vector of structures, and the vector itself internally is going to be called a tab. So, RNTAB is the name of, is the name of the structure which contains this whole thing. And in the main program, rntab will be the name of the actual object. But inside this will use tab to indicate the actual table which contains the roll number name pairs. So, what is the rn structure, well, it is just simply is a roll number name pair. And first we have the construction code over here.

So, what happens over here? We are simply going to read from this stream rnfile which has been passed to the constructor. And we are going to read into an rn type object, small rn so we will read in the roll number and will read in the name and we discussed this earlier, we are going to assume over here that the names a single word for simplicity and the multi word name aspect we will worry about a little bit, we will worry about in the exercise. And let me just point out right now, the multi word names you will have to use getline rather than the input the input redirection operator.

So, we are going to get the roll number and if the file ended we then break. We have done the whole thing otherwise we get the name and then the name and the roll number are stored in this structure rn, we just push back this structure at the end of a vector. So, we keep reading lines and we construct these rn objects and we keep pushing them on our vector we is tab. So, that is how we initialize or we construct this table.

(Refer Slide Time: 26:18)



```
  string rollno, name;
};

struct RNTAB{
  vector<RN> tab;
  RNTAB(ifstream &rnfile){
    while(true){
      RN rn;
      rnfile >> rn.rollno;
      if(rnfile.eof()) break;
      rnfile >> rn.name;      // Assume name is single word for simplicity
                              // Multi word: exercise
                              // Use getline.
      tab.push_back(rn);
    }
  }
  string lookup(string rOrN){
    for(size_t i=0; i<tab.size(); i++){
      if(tab[i].name == rOrN) return tab[i].rollno;
      if(tab[i].rollno == rOrN) return tab[i].rollno;
    }
    return "";
  }
};

int main(int argc, char**argv){
  ifstream rnfile(argv[1]), rsmfile(argv[2]);
  RNTAB rntab(rnfile);
-UU-:----F1  marks1.cpp     56% L83    (C++/l Abbrev) 11:23AM 0.36 ------------
```

Then there is the lookup part. So, how does the lookup part work? So, it is sent a roll number or a name, and we are going to go through the entire table, or the entire vector and we are going to check is the name equal to the roll number, if it is then we return this, the corresponding roll number. If the roll number is itself supplied, then we just return. So, now, in this we are going through the entire table. When we go through the entire table, if we do not return anything then that means this r or n is not found either as the roll number or as the name. So, in that case we are going to return the empty string. Alright, so that is what this RNTAB is and we have seen how it is implemented. So, we will take a quick break here before we look at the major data structure which is RSMTAB.