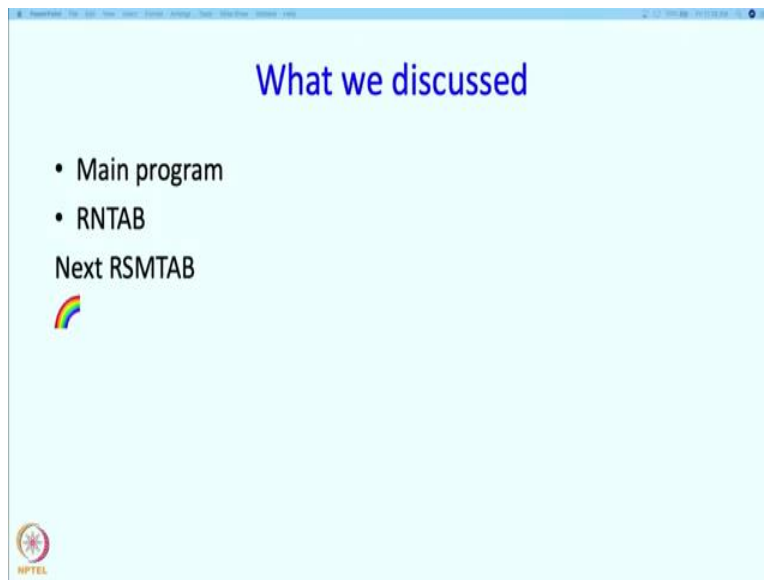


An Introduction to Programming through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Lecture 25: Part 3
Medium Size Programs
RSMTAB and rest of the program

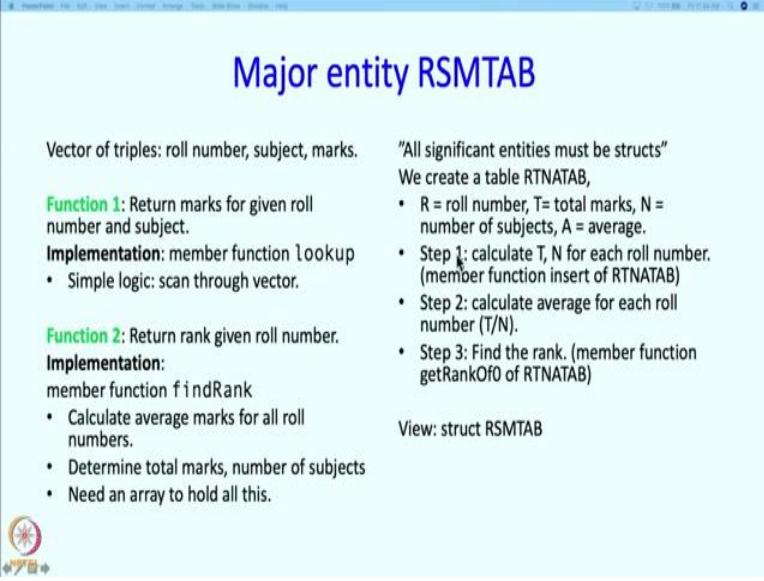
Welcome back!

(Refer Slide Time: 0:28)



In the last segments, we have been talking about the manual program the program inspired by the manual algorithm and we have already discussed the main program and the RNTAB, the table which gives us the translation from the name to the roll number.

(Refer Slide Time: 0:47)



Major entity RSMTAB

Vector of triples: roll number, subject, marks.

Function 1: Return marks for given roll number and subject.
Implementation: member function lookup

- Simple logic: scan through vector.

Function 2: Return rank given roll number.
Implementation: member function findRank

- Calculate average marks for all roll numbers.
- Determine total marks, number of subjects
- Need an array to hold all this.

"All significant entities must be structs"
We create a table RTNATAB,

- R = roll number, T= total marks, N = number of subjects, A = average.
- Step 1: calculate T, N for each roll number. (member function insert of RTNATAB)
- Step 2: calculate average for each roll number (T/N).
- Step 3: Find the rank. (member function getRankOf0 of RTNATAB)

View: struct RSMTAB

In this segment, we will talk about the RSMTAB, the table which contains information about the marks obtained by students with the given roll number in the given subject. So this is the second major entity in our program after the RNTAB. So, this is a vector, and it stores triples which consists of roll number, subject and marks. The first function that this table or this entity or this struct is going to perform is that it will return the marks given the roll number and the subject.

The implementation is going to happen in the member function lookup. So, I used the same name lookup here, you should not be confused with the lookup in RNTAB. Here the lookup is we are looking of the marks; given the roll number and the subject. Over there, we were looking up the roll number given the name. Alright, so the logic of this is fairly straight forward.

We are going to scan through the vector and we are going to look for entry in the vector where the roll number is in the entry is the same as the roll number in the command and the subject in the entry is also the same as the subject in the command and if we have a match, then we just report the marks. The second function is returning the rank given the roll number. Now this is a little bit more complicated because at first we need to calculate the rank. The rank is not immediately available to us.

So, the implementation is done by a member function called findRank. So the way we calculate the rank is we first calculate the average marks for all roll numbers. Again what do I mean by this? So, a particular roll number or a student with a particular roll number will have taken several subjects potentially. So, we are going to look at the marks obtained by that student or by that roll number in all those subjects and we are going to take the average of all of those.

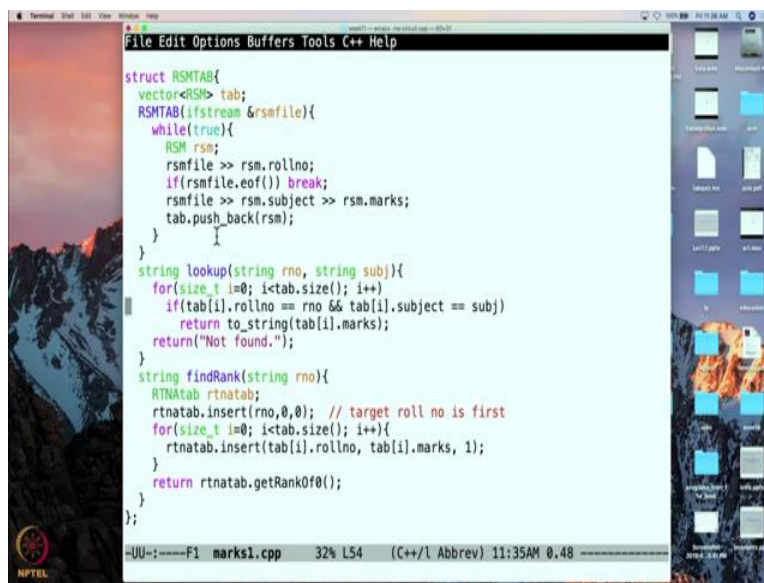
So that is the first step. So, for doing that for calculating the average, we have to first calculate the total number of marks, we have to find out how many subjects there are, how many subjects this particular student has taken, and then we just divide the total marks by the number of subjects. We have to do this for all students because only then we will be able to compare the students. So we need an array to hold this and our principle is that all significant entities must be structs. So, we will create another table, let us call it RTNATAB where R denotes roll number, T denotes total marks, N is the number of subjects and A is the average. So, we are going to use the table to calculate the average marks for a given roll number and how does this happen? In the step 1 we will calculate T and N. So, this is done by the member function insert. So, we will see this member function insert. So, it will go over RSMTAB and it will look at triples of this kind and for every roll number, subject, mark triple, it will go to RTNATAB for that corresponding roll number and it will add the marks into this total. And we do not really need to know the name of the subject now, we just need to know how many subjects the student is taking.

So, since, we just added the marks for one subjects into T will increment N the number of subjects by 1. So this way, you will keep accumulating the total marks and the total number of subjects taken by the students as we scan through RSMTAB. Then in the second step, we are going to calculate the average for each roll number. So, we are going to update this A field. So, A is the average field where A is the average. So that is what we are going to update and that update is simple. So, we already know the total marks obtained by the student from step 1 and we already know the number of subjects taken by the students, so we just try to take the ratio T by N. So, T by N will be stored in A and then we have this RTNATAB, which contains the average marks for a given roll number

and now we have to find the rank.

How do we do this? Well, we look at our students and will simply count how many other students there are whose average is higher and the average is already been calculated. So, for this, we again we just have to scan through this RTNATAB once. So, we will see the details in a minute. So, now we are going to view the structure RSMTAB and we will see, how the details of how these functions are being performed, so first we will look at this lookup and then we will look at findRank.

(Refer Slide Time: 7:09)

A screenshot of a C++ code editor window titled 'marks1.cpp'. The code defines a structure RSMTAB containing a vector of RSM objects. It includes a while loop to read data from a file into the RSMTAB structure. Below this, there are two functions: 'lookup' which searches for a student by roll number and subject, and 'findRank' which calculates the rank of a student by comparing their marks with others in the RSMTAB. The status bar at the bottom shows 'F1 marks1.cpp 32% L54 (C++/1 Abbrev) 11:35AM 0.48'.

```
struct RSMTAB{
    vector<RSM> tab;
    RSMTAB(ifstream &rsmfile){
        while(true){
            RSM rsm;
            rsmfile >> rsm.rollno;
            if(rsmfile.eof()) break;
            rsmfile >> rsm.subject >> rsm.marks;
            tab.push_back(rsm);
        }
    }

    string lookup(string rno, string subj){
        for(size_t i=0; i<tab.size(); i++){
            if(tab[i].rollno == rno && tab[i].subject == subj)
                return to_string(tab[i].marks);
            return("Not found.");
        }
    }

    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();
    }
};
```

So, we are going to look at this RSMTAB. So, let us look at the RSMTAB. So this is our structure RSMTAB. As I said, it is performing two functions, and the first function is the lookup function and the second function is the findRank function. The lookup function requires a roll number and the subject and the idea is actually fairly straightforward what you might guess.

We are going to go, we are going to go through all the entries in this RSMTAB. So, the all the entries in the RSMTAB. So let us look at this structure RSMTAB so what does it contain? It contains a vector of struct RSM and the vector we are going to call tab or table in this in this code. So, let us look at what this vector the vector elements are.

(Refer Slide Time: 8:06)

nicest thing to return would be a message not found. So, therefore, we have chosen the type of this lookup function to be string and so here we are returning the marks by first converting to the string and if the entry, appropriate entry is not found, then we will return the string not found so that is the lookup which is fairly easy.

The more complicated thing is the rank find, is the findRank function, findRank member function. So, here we are given the roll number and we want to find the rank of the student having this roll number. As we said earlier, we need an additional local data structure called RTNATAB, so this RTNATAB is going to contain all the data associated with all the students.

So. it will contain the, it is the structure that we are going to use to calculate the average and the way this works is we are going to insert we are going to go over the file and we are going to insert the information about students and roll numbers and marks so let us look at this RTNATAB.

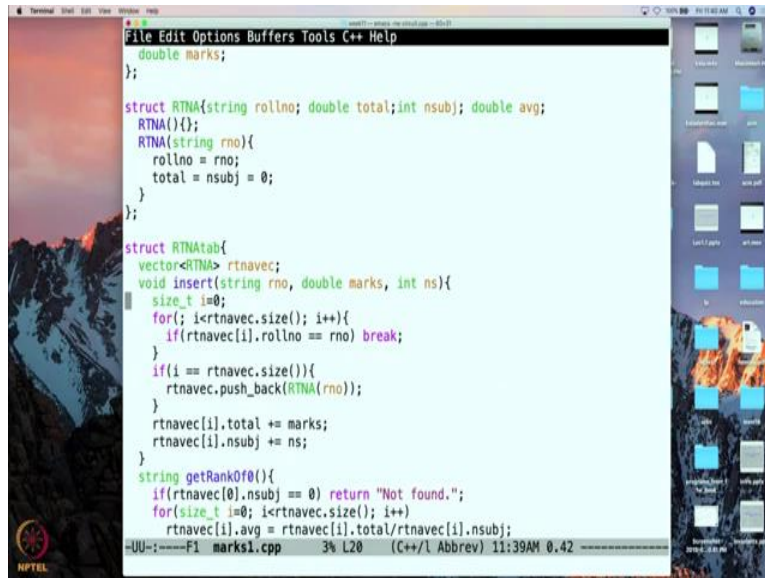
(Refer Slide Time: 11:28)



```
};

struct RSMTAB{
    vector<RSM> tab;
    RSMTAB(ifstream &rsmfile){
        while(true){
            RSM rsm;
            rsmfile >> rsm.rollno;
            if(rsmfile.eof()) break;
            rsmfile >> rsm.subject >> rsm.marks;
            tab.push_back(rsm);
        }
    }
    string lookup(string rno, string subj){
        for(size_t i=0; i<tab.size(); i++){
            if(tab[i].rollno == rno && tab[i].subject == subj)
                return to_string(tab[i].marks);
            return("Not found.");
        }
    }
    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();
    }
};
```

The screenshot shows a C++ code editor with a menu bar (File, Edit, Options, Buffers, Tools, C++, Help) and a toolbar. The code defines a struct RSMTAB with a vector of RSM objects and two methods: lookup and findRank. The lookup method searches for a student by roll number and subject, returning their marks as a string or "Not found.". The findRank method uses an RTNATAB (Rank Table) to calculate the rank of a student based on their roll number. The status bar at the bottom indicates the file is 'marks1.cpp', line 32, column 153, in C++ mode, at 11:39 AM on 0.42.

A screenshot of a C++ code editor window. The code defines a structure RTNA with fields rollno, total, nsubj, and avg. It also defines a structure RTNATAB containing a vector of RTNA objects. The RTNATAB structure has an insert method that takes a roll number, marks, and number of subjects, and a getRankOf method that returns the rank of a given roll number. The code is written in C++ and uses standard library containers like vector and string.

```
double marks;
};

struct RTNA{string rollno; double total;int nsubj; double avg;
RTNA();
RTNA(string rno){
rollno = rno;
total = nsubj = 0;
}
};

struct RTNATAB{
vector<RTNA> rtnavec;
void insert(string rno, double marks, int ns){
size_t i=0;
for(; i<rtnavec.size(); i++){
if(rtnavec[i].rollno == rno) break;
}
if(i == rtnavec.size()){
rtnavec.push_back(RTNA(rno));
}
rtnavec[i].total += marks;
rtnavec[i].nsubj += ns;
}

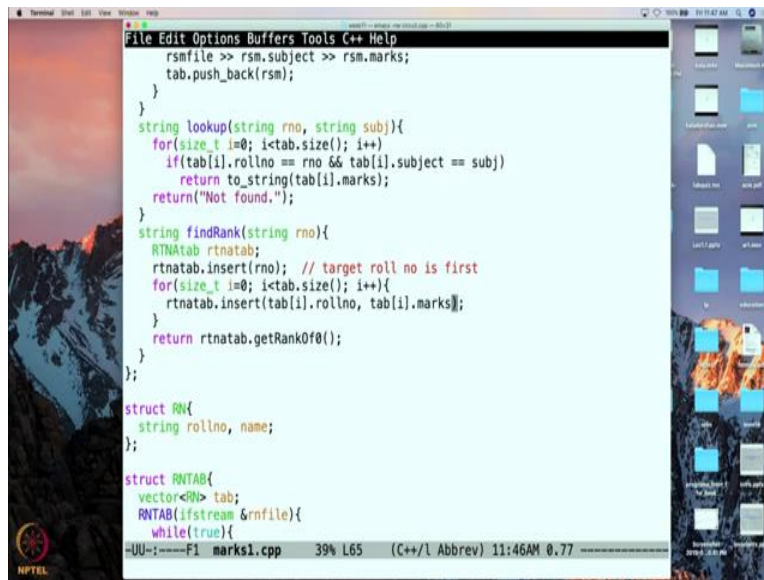
string getRankOf0(){
if(rtnavec[0].nsubj == 0) return "Not found.";
for(size_t i=0; i<rtnavec.size(); i++)
rtnavec[i].avg = rtnavec[i].total/rtnavec[i].nsubj;
}
};
```

So, RTNATAB is as we said earlier, a data structure itself consisting of structures of a certain type. So, let us look at that. So, it contains structures of the type RTNA so it contains roll number, a total number of subjects and average. So, how does it get built up? Well, initially it starts of empty and now we have provided an insert member function, so what does the insert member function do?

It takes the roll number and the marks obtained by the student, and it says and usually we will just call it with the marks obtained in one subject, but occasionally, we want to call it without marks obtained in any subject, so we just want to say that look this roll number exists and do something about it, so make an entry corresponding to that roll number.

So, that is why we are going to have, we are going to have two arguments over here and the argument is going to be double with marks equal to 0 and the number of subjects equal to 0. We will finish discussing this lookup function.

(Refer Slide Time: 13:03)

A screenshot of a C++ IDE window titled 'marks1.cpp'. The code defines a 'lookup' function that iterates through a table to find a subject by roll number and subject name. It also defines a 'findRank' function that uses a local 'RTNATAB' structure to find the rank of a given roll number. The 'RTNATAB' struct contains a vector of 'RTNAB' structs. The 'RTNAB' struct contains 'rollno' and 'name' fields. The 'findRank' function inserts the target roll number into the 'RTNATAB' and then iterates through it to find the rank. The status bar at the bottom shows '39% L65 (C++/l Abbrev) 11:46AM 0.77'.

```
File Edit Options Buffers Tools C++ Help
rsmfile >> rsm.subject >> rsm.marks;
tab.push_back(rsm);
}
string lookup(string rno, string subj){
    for(size_t i=0; i<tab.size(); i++){
        if(tab[i].rollno == rno && tab[i].subject == subj)
            return to_string(tab[i].marks);
        return("Not found.");
    }
}
string findRank(string rno){
    RTNATAB rtnatab;
    rtnatab.insert(rno); // target roll no is first
    for(size_t i=0; i<tab.size(); i++){
        rtnatab.insert(tab[i].rollno, tab[i].marks);
    }
    return rtnatab.getRankOf0();
}
};

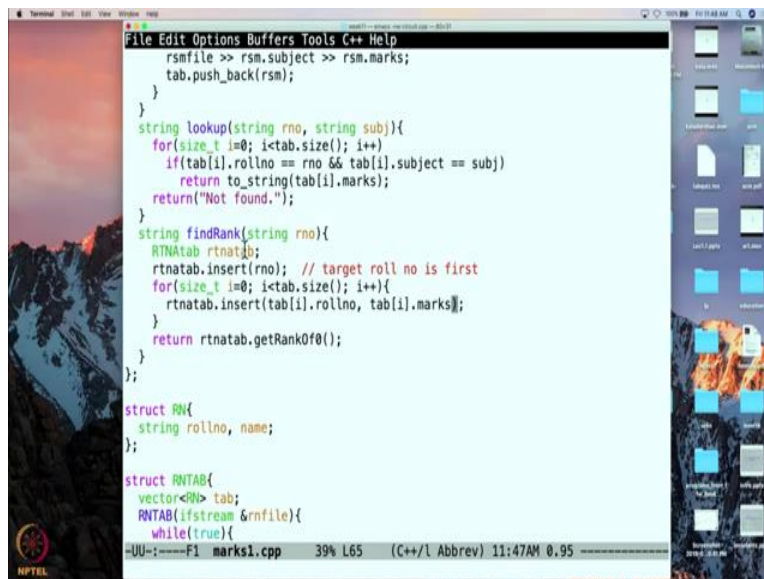
struct RTNAB{
    string rollno, name;
};

struct RTNATAB{
    vector<RTNAB> tab;
    RTNATAB(ifstream &rsmfile){
        while(true){
            rsmfile >> rsm.subject >> rsm.marks;
            tab.push_back(rsm);
        }
    }
};

-|JU-1-====F1 marks1.cpp 39% L65 (C++/l Abbrev) 11:46AM 0.77
```

Now we are going to discuss the findRank member function. So, let me remind you the findRank member function is going to figure out the rank of the given roll number. So, for that, it has to calculate the average marks obtained by all students and then see where this roll number is in that list of marks, so to do that, we will need a local data structure called RTNATAB, so let us first look at this RTNATAB and then we will see how we are going to use it.

(Refer Slide Time: 13:44)

A screenshot of a C++ IDE window titled 'marks1.cpp'. The code defines a 'lookup' function that iterates through a table to find a subject by roll number and subject name. It also defines a 'findRank' function that uses a local 'RTNATAB' structure to find the rank of a given roll number. The 'RTNATAB' struct contains a vector of 'RTNAB' structs. The 'RTNAB' struct contains 'rollno' and 'name' fields. The 'findRank' function inserts the target roll number into the 'RTNATAB' and then iterates through it to find the rank. The status bar at the bottom shows '39% L65 (C++/l Abbrev) 11:47AM 0.95'.

```
File Edit Options Buffers Tools C++ Help
rsmfile >> rsm.subject >> rsm.marks;
tab.push_back(rsm);
}
string lookup(string rno, string subj){
    for(size_t i=0; i<tab.size(); i++){
        if(tab[i].rollno == rno && tab[i].subject == subj)
            return to_string(tab[i].marks);
        return("Not found.");
    }
}
string findRank(string rno){
    RTNATAB rtnatab;
    rtnatab.insert(rno); // target roll no is first
    for(size_t i=0; i<tab.size(); i++){
        rtnatab.insert(tab[i].rollno, tab[i].marks);
    }
    return rtnatab.getRankOf0();
}
};

struct RTNAB{
    string rollno, name;
};

struct RTNATAB{
    vector<RTNAB> tab;
    RTNATAB(ifstream &rsmfile){
        while(true){
            rsmfile >> rsm.subject >> rsm.marks;
            tab.push_back(rsm);
        }
    }
};

-|JU-1-====F1 marks1.cpp 39% L65 (C++/l Abbrev) 11:47AM 0.95
```

So, the struct RTNATAB contains a vector of structs RTNAB, so as we have discussed this

earlier, RTNA is simply a quadruple, so it contains roll number, total, number of subjects seen so far and the average, so the purpose of this is to find out for each roll number, what is the total number of marks, how many subjects there are and then once we have built up these two fields, we can just calculate the average by dividing this by this, so here are the constructors for it. So, there are two constructors, so we can create either a dummy this is the default constructor and this will construct an RTNA for a given roll number, so the roll number field, the roll number member will be given a certain roll number, but at this point we are just starting of this record in our structure that we are going to keep and therefore, the total and the number of subjects will all be set to zero.

So, let us come to this RTNATAB, so we started off with an, we started off with an empty RTNATAB and then we have two insertions possible into it. So, this insertion just inserts the roll number, so if you are just inserting a roll number, this is what we need to do before we insert any marks. So at that point, we are going to push back the roll number and the empty data structure onto that roll number. So, basically rtnavec will contain just the roll number and the other fields will be 0 and then there is the other insert function.

(Refer Slide Time: 15:35)



```
};

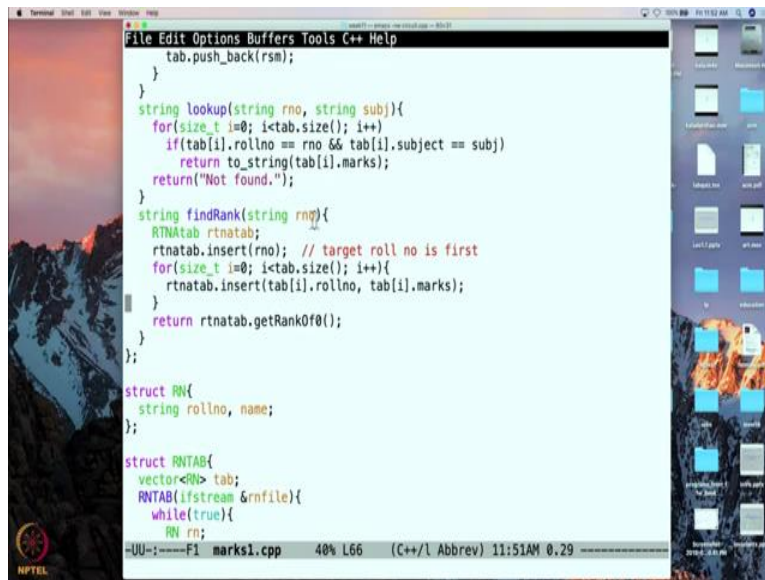
struct RTNatab{
    vector<RTNA> rtnavec;
    void insert(string rno){
        rtnavec.push_back(RTNA(rno));
    }
    void insert(string rno, double marks){
        size_t i=0;
        for(; i<rtnavec.size(); i++){
            if(rtnavec[i].rollno == rno) break;
        }
        if(i == rtnavec.size()){
            rtnavec.push_back(RTNA(rno));
        }
        rtnavec[i].total += marks;
        rtnavec[i].nsubj ++;
    }
    string getRankOf0(){
        if(rtnavec[0].nsubj == 0) return "Not found.";
        for(size_t i=0; i<rtnavec.size(); i++){
            rtnavec[i].avg = rtnavec[i].total/rtnavec[i].nsubj;
            int rank = 1;
            for(size_t i=1; i<rtnavec.size(); i++){
                if(rtnavec[i].avg > rtnavec[0].avg) rank++;
            }
            return to_string(rank);
        }
    }
};
```

marks1.cpp 9% L28 (C++/l Abbrev) 11:49AM 0.30

So, in this case we are going to not, we are going to insert the marks for the roll number, so how does that work? Well, we are going to go through this table, this rtnavec and in that rtnavec, we are going to check whether the roll number appears, so if the roll number

appears, then we want to know at what position it appears. So, we have discovered this `i`, so we break over here and if we do not break, if we go to the end then `i` will be equal to `rtnavec` size. So, if it is `rtnavec` size as we check over here then we know that this roll number is not present in the vector. So, we insert it into the vector. So for this, we push back this RTNA of roll number. So, this is exactly what we did over here. And now we have `i` pointing to the index where that roll number appears in `rtnavec`, either it appear, it was already there or it appeared because we pushed it at the last position. So, once we know that we simply have to add the marks to the total which is being accumulated there and we have to increment `N` subjects, the number of subjects have to be incremented, so that finishes this function of insertion, so let us see how this function gets used in our `findRank` function.

(Refer Slide Time: 17:20)



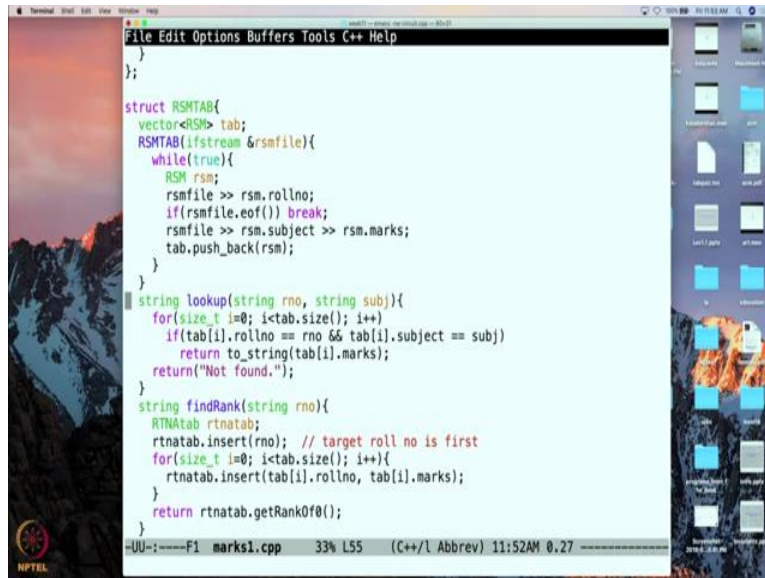
```
File Edit Options Buffers Tools C++ Help
tab.push_back(rsm);
}
}
string lookup(string rno, string subj){
    for(size_t i=0; i<tab.size(); i++){
        if(tab[i].rollno == rno && tab[i].subject == subj)
            return to_string(tab[i].marks);
        return("Not found.");
    }
}
string findRank(string rno){
    RTNatab rtnatab;
    rtnatab.insert(rno); // target roll no is first
    for(size_t i=0; i<tab.size(); i++){
        rtnatab.insert(tab[i].rollno, tab[i].marks);
    }
    return rtnatab.getRankOf0();
};

struct RN{
    string rollno, name;
};

struct RNTAB{
    vector<RN> tab;
    RNTAB(lifstream &rfile){
        while(true){
            RN rn;
            if(!rfile.getline(rn.rollno, rn.name, '\n'))
                break;
            tab.push_back(rn);
        }
    }
};

int main(){
    RNTAB rntab(lifstream("marks1.txt"));
    string rno, subj;
    cout << "Enter roll number and subject: ";
    getline(cin, rno);
    getline(cin, subj);
    string marks = lookup(rno, subj);
    cout << "Marks: " << marks << endl;
    string rank = findRank(rno);
    cout << "Rank: " << rank << endl;
    return 0;
}
```

NPTEL



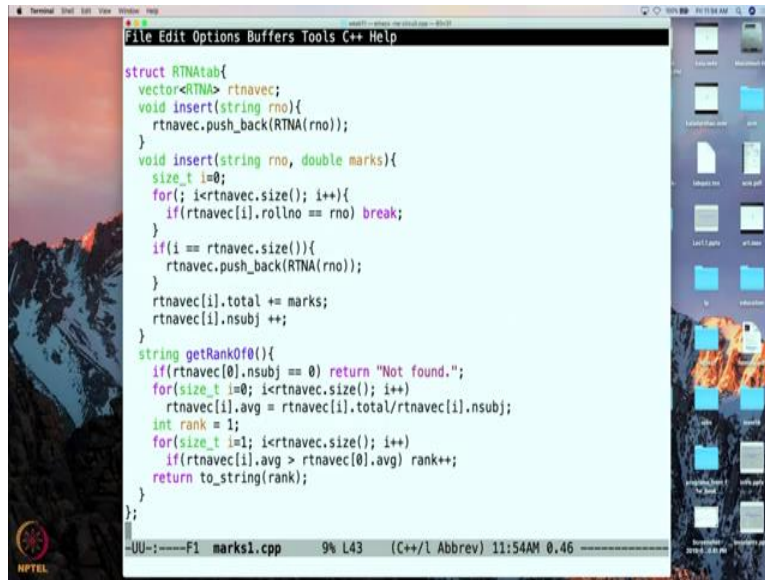
```
};

struct RSMTAB{
    vector<RSM> tab;
    RSMTAB(ifstream &rsmfile){
        while(true){
            RSM rsm;
            rsmfile >> rsm.rollno;
            if(rsmfile.eof()) break;
            rsmfile >> rsm.subject >> rsm.marks;
            tab.push_back(rsm);
        }
    }
    string lookup(string rno, string subj){
        for(size_t i=0; i<tab.size(); i++){
            if(tab[i].rollno == rno && tab[i].subject == subj)
                return to_string(tab[i].marks);
            return("Not found.");
        }
    }
    string findRank(string rno){
        RTNATAB rtnatab;
        rtnatab.insert(rno); // target roll no is first
        for(size_t i=0; i<tab.size(); i++){
            rtnatab.insert(tab[i].rollno, tab[i].marks);
        }
        return rtnatab.getRankOf0();
    }
};
```

So here is the findRank function. So, what are we going to do? Well at position 0 or at the very beginning, we make an entry for the roll number whose rank we want to determine, because we are going to quickly need the average marks for this roll number and so we, so let us just place it at position 0. So, I have written it first so I really mean zeroth because we are counting from 0 over here, so we inserted it at roll number 0.

Then we went through our table, our RSM table tab refers to that RSM table and we took the roll numbers from it and so we looked at the size of our RSM table. So, remember all this is in the context of our RSMTAB, so let me just position it properly so that you can see what this tab refers to. So, this tab is referring to this RSMTAB, so if you remember RSMTAB consists of roll number, subject code and marks. So, we are going to go through all the entries. So tab[i].roll number is the roll number of ith entry. We are not concerned with the subject name but we are concerned with the marks. So, the ith entry tells us that roll number tab[i].roll number has got tab[i].marks in some subjects and since, we want to know what is the total number of marks that this roll number has got, we are going to insert this into RTNATAB. So, this way we end up inserting all the marks got by all the roll numbers into this RTNATAB and RTNATAB at this point will include for every roll number what is the total number of marks and what is the total number of subjects taken by that roll number. So we are just about ready to calculate the average and what we do is we will do it directly. We will just call RTNATAB of getRankOf0().

(Refer Slide Time: 19:45)



```
File Edit Options Buffers Tools C++ Help
struct RTNATAB{
    vector<RTNA> rtnavec;
    void insert(string rno){
        rtnavec.push_back(RTNA(rno));
    }
    void insert(string rno, double marks){
        size_t i=0;
        for(; i<rtnavec.size(); i++){
            if(rtnavec[i].rollno == rno) break;
        }
        if(i == rtnavec.size()){
            rtnavec.push_back(RTNA(rno));
        }
        rtnavec[i].total += marks;
        rtnavec[i].nsubj ++;
    }
    string getRankOf0(){
        if(rtnavec[0].nsubj == 0) return "Not found.";
        for(size_t i=0; i<rtnavec.size(); i++){
            rtnavec[i].avg = rtnavec[i].total/rtnavec[i].nsubj;
        }
        int rank = 1;
        for(size_t i=1; i<rtnavec.size(); i++){
            if(rtnavec[i].avg > rtnavec[0].avg) rank++;
        }
        return to_string(rank);
    }
};

--UU--F1 marks1.cpp 9% L43 (C++/l Abbrev) 11:54AM 0.46
```

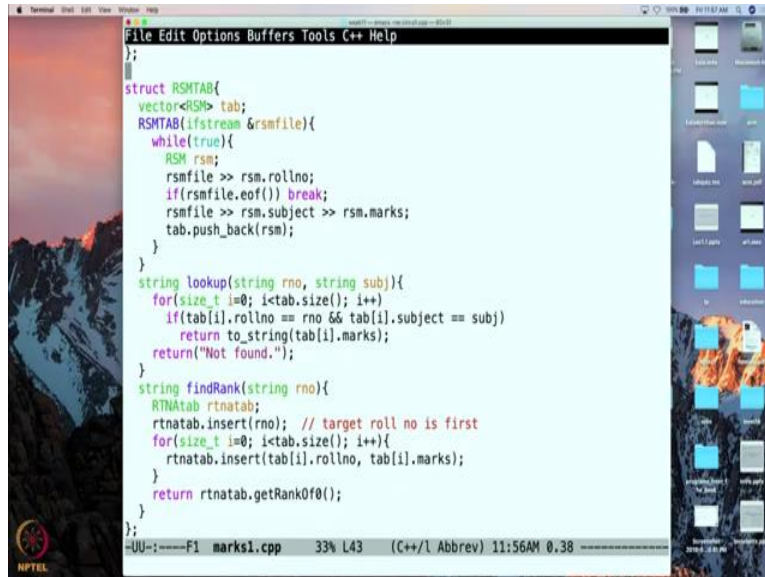
So let us go back and see that, so getRankOf0() is going to start off with that roll number, so that we well, we have not said what the roll number is but the roll number in the zeroth position. Now, in the zeroth position did we insert any subjects, well, we may or may not have, so we are going to have to check, so if the n subjects is 0, then in going through this like entire RSMTAB, we did not find the given roll number, so we are immediately going to return not found.

Otherwise, first we are going to calculate the average, so here is the average being calculated. So, for all the elements in the table we are just going to set rtnavec[i].average equals the total divided by the number of subjects. So, at this point, when we come to this point of the code rtnavec contains the average marks obtained by every student and now all the remains is to compare the marks of the zeroth entry, the entry the roll number that we are interested in which we cleverly stored in the zeroth position. So we are going to we are going to get that entry and we are going to compare the marks of that entry with all other entries, so we are going to go over the entire table starting at i equal to 1 and if we, wherever we find that the marks of some other roll numbers are larger than the marks of our roll number, then we are going to increment rank by 1.

And we started off with rank equal to 1, so that at the end of it whenever we find the higher average marks, our rank will be incremented eventually we will get the

appropriate rank and again we are going to convert it into string and return it back because we potentially also want to return not found over here.

(Refer Slide Time: 22:16)



```
};
}

struct RSMTAB{
    vector<RSM> tab;
    RSMTAB(ifstream &rsmfile){
        while(true){
            RSM rsm;
            rsmfile >> rsm.rollno;
            if(rsmfile.eof()) break;
            rsmfile >> rsm.subject >> rsm.marks;
            tab.push_back(rsm);
        }
    }

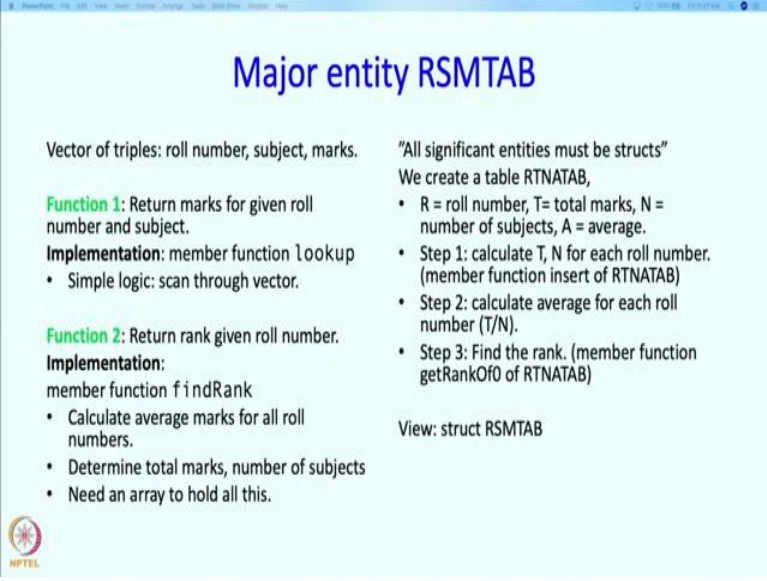
    string lookup(string rno, string subj){
        for(size_t i=0; i<tab.size(); i++){
            if(tab[i].rollno == rno && tab[i].subject == subj)
                return to_string(tab[i].marks);
            return("Not found.");
        }
    }

    string findRank(string rno){
        RTNATAB rtnatab;
        rtnatab.insert(rno); // target roll no is first
        for(size_t i=0; i<tab.size(); i++){
            rtnatab.insert(tab[i].rollno, tab[i].marks);
        }
        return rtnatab.getRankOf0();
    }
};

//UJ-:====F1 marks1.cpp 33% L43 (C++/l Abbrev) 11:56AM 0.38
```

Alright, so that completes the description of this RSMTAB and internally it used this RTNATAB as well.

(Refer Slide Time: 22:42)



Major entity RSMTAB

Vector of triples: roll number, subject, marks.

Function 1: Return marks for given roll number and subject.
Implementation: member function lookup

- Simple logic: scan through vector.


Function 2: Return rank given roll number.
Implementation: member function findRank

- Calculate average marks for all roll numbers.
- Determine total marks, number of subjects
- Need an array to hold all this.

"All significant entities must be structs"
We create a table RTNATAB,

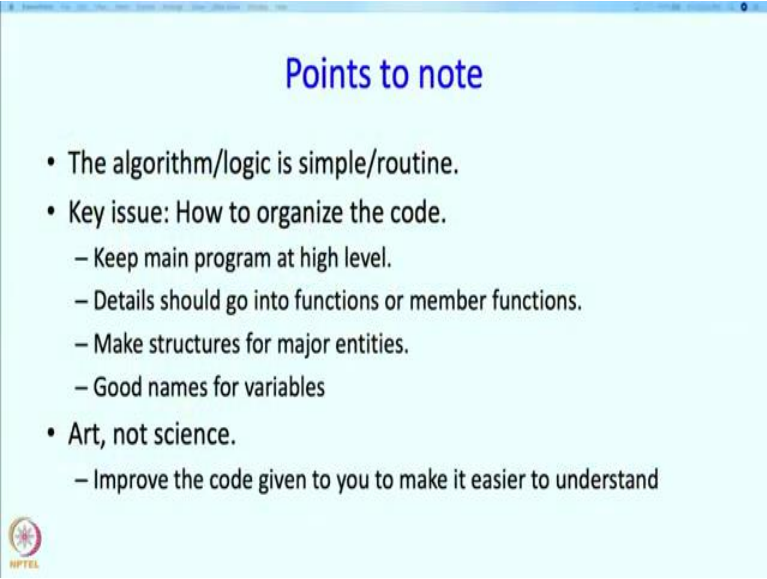
- R = roll number, T= total marks, N = number of subjects, A = average.
- Step 1: calculate T, N for each roll number. (member function insert of RTNATAB)
- Step 2: calculate average for each roll number (T/N).
- Step 3: Find the rank. (member function getRankOfO of RTNATAB)

View: struct RSMTAB




So, let us return back to the presentation.

(Refer Slide Time: 22:50)



Points to note

- The algorithm/logic is simple/routine.
- Key issue: How to organize the code.
 - Keep main program at high level.
 - Details should go into functions or member functions.
 - Make structures for major entities.
 - Good names for variables
- Art, not science.
 - Improve the code given to you to make it easier to understand

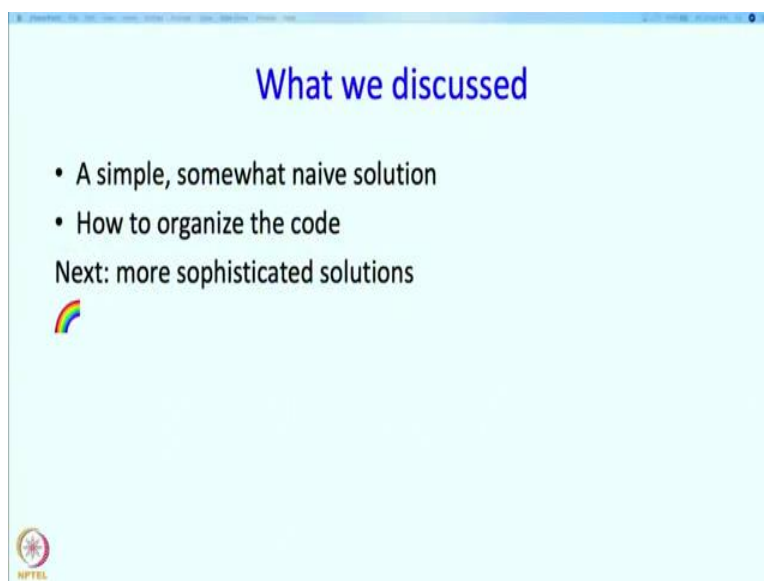


So, there are number of points to note in what we have seen. In some sense, the algorithm or the logic is quite simple or routine. I mean there are lots of things to do, but what we are doing is not particularly clever. We are calculating averages, we are finding out how many averages are bigger than the average of the roll number of our interest, so a lot of things to be done, but the logic itself is fairly straight forwarded.

The key issue is how do we organize this so that the code looks easy to understand, so the issues has always are keep the main program at high level, details should go into functions and member functions and we have to make structures for major entities. So the major entity from the point of view of the main program was RNTAB and also RSMTAB but while processing RSMTAB, while calculating the average marks, we had to create another entity called RTNATAB.

So this was, this again we made, for this again we made a structure and we have to try and give good names for variables and at the end of it, I have to say that this is an art and not science and you may not like every step that I have taken; you may say that, “Oh, I think I can write this step better”, and I will invite you to do that and this is an art in the sense that what you like, I may not like. But the point is you still have to appreciate these points, that you have to appreciate that code has to be understandable and at least according to your sense of what is easy or what is easy to understand or what is elegant you should be developing the code and you should be developing this sense of what is easy and what is easy to understand and it is of course, possible that your sense of what is easy might be better than my sense of what is easy and I will definitely invite you to improve the code that has been given and make it more easy to understand and that is certainly an ongoing process.

(Refer Slide Time: 25:41)



Alright, so what have we discussed in all of this? We have discussed a simple, somewhat naive solution. It is simple and naive in the sense that it is sort of repeats computation and we have seen how the code for this can be organized. Next, we are going to see some sophisticated solutions. So we will take a break.