Computational Chemistry & Classical Molecular Dynamics

Prof. B. L. Tembe
Department of Chemistry
Indian Institute of Technology- Bombay

Lecture – 05
Programming Techniques 3. Roots of a Quadratic Equation and Arrays

Hello and welcome again to this course, computational chemistry and classical molecular dynamics, so, so for what we have been doing is to consider several types of program statements, so we had different types of statements such as a conditional statement, execution, executable statement, do statement which is a loop statement and we started this program to calculate the roots of a quadratic equation.

So, you must have seen from the last class that the if statements were used where other complex if statements, so many conditions were satisfied, so if one condition is satisfied, do something if something else is satisfied, do something and finally, if none of them is satisfied that end that if statements, so these are complex if structures but if you want a very simple if statement like suppose, I want the absolute value of a number.

How do I get an absolute value? If the number is positive, the absolute value is the same as the number and if the number is negative, then the absolute value is minus that number, so the if statement will read like this, if into bracket x. greater .0.0 bracket complete, absolute x = x, so that is if x is > 0, then the absolute value of x is x itself.

(Refer Slide Time: 01:52)

program quadratic

```
c    program quadratic
     write (*, *) 'input the values of a, b and c:'
     read (*, *) a, b, c
     if ((a .eq. 0.0) .and. (b .ne. 0.0)) then
     x = -c/b
      twoa=2.0*a
     write (*, *) 'the solution of linear equation x=', x
     go to 100
     else if ((a .eq. 0.0) .and. (b .eq. 0.0)) then
     write (*, *) 'both coefficients a and b are zero'
     go to 100
     endif
```

Then the next if statement would be if into bracket x.lt.0 bracket complete that is if x < 0 then the absolute value of x is –x, so then you say, if into bracket x.lt.0 bracket complete absolute x = -x, so in one line itself you can complete the work of an if statement. Now, in our quadratic equation we have to satisfy many, many conditions to solve the problem for example, you look at this program, we started looking at it last time.

So, we input the values of a, b and c that is the first part of a quadratic equation that is your quadratic equation is ax square + bx + c = 0, so you first one to ascertain that if this a = 0, then it is not a quadratic equation anymore, so therefore you have this first condition, if a = 0 and b != 0, so these 2 conditions are satisfied, then you say x = -c/b, okay, so that is how you solve that linear equation.

Then, you want to write that solution that the solution of a linear equation x = x; x itself is the value, so if a = 0 and b!= 0 then this is the solution of the linear equation, you will write on the screen and then go to 100, going to 100 is really saying that it is the end of the program because you have solved the problem. Then, once the first condition suppose, it is not satisfied then only you will go to the second statement.

Because if a = 0 and b!= 0, then you have done all these operations then you will go to 100, you will never come to this second part of the if statement, if a is greater; a = 0 and b!= 0 but now,

suppose b is also = 0, so you will come to this second part of the if statement, if a = 0 and b is also = 0. So, if both these are 0, then there is no equation, so we just print that both coefficients a and b are = 0 and then you again exit, go to 100, go to 100 is the end of that program.

And then you will end if, now suppose both these conditions are not satisfied, so if both these are not satisfied, then that value of a is not 0, then you will go to the next line. So, you will go to the next line after this end if statement only if a! = 0, so that means it is a real quadratic equation.

(Refer Slide Time: 04:41)

```
      ww = b * b - 4.0 * a * c
     if (ww .lt. 0.0) then
     go to 50
     else
     rtofww = sqrt (ww)
     root1 =( -b + rtofww) / twoa
     root2 =( -b - rtofww) / twoa
     write (*, *) 'real roots 1 and 2 are', root1, root2
     go to 100
     endif
 50  continue
c    the roots are complex b**2 - 4 * a * c is -ve
     ww = 4.0 * a * c - b * b
     rtofww = sqrt (ww)
     realpt = -b / twoa
     ximpt = rtofww / twoa
     write (*, *) 'complex roots'
     write (*, *) 'root1', 'real part=', realpt, 'imaginary part=', ximpt
      ximpt2 = -ximpt
      write (*, *) 'root2', 'real part=', realpt, 'imaginary part=', ximpt2
100  continue
     end
```

So, all these statement that we wrote is just to make sure that you have a proper quadratic equation and the reason to do that is that whenever you feed data, data could be anything, a can be 0, so therefore your program should be written in such a way that whatever your data, it should take care of it that is why you have to write so many lines here although, you know that the solution of a quadratic equation is just one simple square root divided by 2a.

So, now we come to this part, when we come here, we know that a is not 0 and b could be 0 here, does not matter but a is not 0 that means, it is a quadratic equation, so then you compute this, b square – 4ac, this is the calculation of b square – 4 ac and if this b square – 4 ac is < 0, then you go to this 50; line 50. So, b square -4 ac is > 0, then you will come to the this set of part, so that your new if statement; there are 2 parts if something is <0, then do all these, then go to 50.

Else, you do all these things, then go to 100, okay, so this is the second if statement, it has again 2 components; one component if s is; if w < 0, then you do this part, if it is not less; if it is not <0, then you do these things. What are these lines now? Rtofww that is square root of ww, so this line calculates square root of ww, the next line calculate the first root; what is the first root; -b + that square root of b square – 4 ac divided by 2a.

In the earlier slide, you already saw that 2a is 2 times a, so this is my first root, similarly this is my second root. What is the second root? -b – square root of b square – 4 ac divided by 2a, so these are my 2 roots, I just write those 2 roots and say that these are real root 1 and 2 are root 1 and root 2, so you have finish the calculation for real roots, again go to 100, now what is this 100? 100 continue end, so this signifies the end of the program.

Now, in case this ww is < 0, then you will come to this part, then 50 continue; 50 continue means, you just start executing the statements after 50, 50 continue is really a statement which say that you keep going ahead, okay, so whereas a go to 100 statement, it tells you to go to this statement that is continue then end. So, continue statements and go to statements are just allowing you to keep going ahead in the way that is assigned.

So now, I come to this statement, when I come to this particular line several conditions are satisfied that it a != 0 and that b square -4 ac is negative, okay, so now we already know that these roots are complex okay, so this particular line, it is a comment card, so there should be a c at the starting point otherwise, this computer; Fortran compiler will not understand what this line is because this is not an executable statement, this is just an English statement.

The roots are complex, if something is satisfied, so this is not a Fortran statement, there should be a comment card, so just put a c in the first line here, there should be a, c here so now, since ww was negative, so now I will calculate ww as 4 ac – b square, so it is the reverse of that. Now, we will surely know that this is positive because if b square -4 ac is negative, then 4 ac – b square is positive.

So, I have calculate now, ww which is a positive number and calculate the square root of that now, you calculate the square root of 4 ac – b square, again this is a positive number okay, then though we know that the real part is –b/2a, so whether you have real roots or complex roots, the real part is always –b/2 a and if b square -4 ac is positive, then it will add some more part as you have done in the earlier case.

You see here, you have added –b/2a + square root of that so, this is the first root and second root is minus of that so, when I come here, so the real part is always square; -b/2a that is a real part, now the imaginary part, there will be 2, so x imaginary part will be root of fww divided by 2a, okay so this is; remember I have calculated already root of www, so that divided by 2a that is the imaginary part.

So, now your roots are 2; real part + this imaginary part and the second one will be real part - that imaginary part, so there will be a 2 roots; x + iy and x – iy, so what this imaginary part is; just the y part, so once you calculate that y part, then you again write on the screen; write *, * complex roots because when you come up to this part of the program, the roots are already complex.

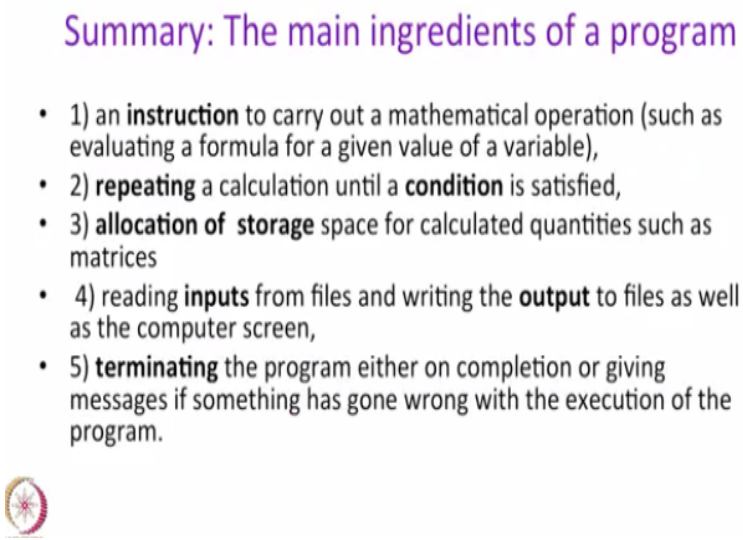
So, once you know their complex, now you write; write *, * root 1, okay, this is again written on the screen, real part is written because real part is –b/2a, then imaginary part = ximpt, so in the first root, it is the real part and + imaginary part and the second root will be the real part –the imaginary part, so I am saying now, x imaginary part 2, this is the second root that second root is the imaginary part is x – y; iy.

So, this –x imaginary part is nothing but –y, so what do I write now, the final root? Write *, * that means, again write on the screen, whatever is in the quotation mark, it will write on the screen, then it will also write the real part then, it will write the real part here which is –b/2a, then imaginary part is minus this, okay.

So, this way you have written the 2 roots from the screen; x + iy and x - iy and once you write that your job is done, it goes to 100 continue end, so you see that with all these lines, these lines

will almost be about 30 or 40 lines, you have taken care of all possibilities, a = 0, a not 0, real solutions and complex solutions and your write statements are such that on the screen, you will know that whether it is a real root or a complex root.

(Refer Slide Time: 11:41)



So, this completes our calculation of the roots of a quadratic equation, okay, then we will proceed to our next topic. So, now again before we go to the next topic let us summarise again what we have done, okay, so let us see what are the main ingredients of a program again, this you have to keep thinking again and again, so that this is the program structure, it will have program structure has an instruction which carries mathematical operation such as evaluating a formula, okay.

Then, repeating a calculation until a condition is satisfied, this is another way of doing, some of the other things we have not seen, allocation of storage, so this we will come now because when you calculate many results, you have to store those things in a memory, so how do you create storage in a program, then reading inputs from files this is what we will be doing now. So far whatever inputs we had you are taking directly from the screen.

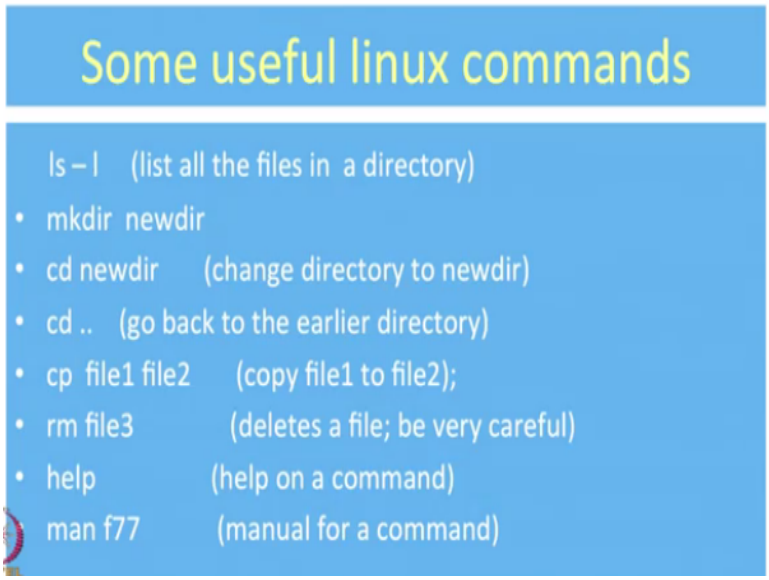
And you are writing the output as well to the screen itself but this is okay if you have 1 number to read or 2 numbers to write, suppose you have 100 numbers to write, so it is very difficult to read 100 numbers on the screen or it is very difficult to give lots of data from the screen to the

computer, it will be too much and suppose, when you are typing the 70th number, you make a mistake, the whole program will crash.

So, therefore the way we read inputs and the way we write output can be done from files rather than the screen, so that we will consider shortly and finally, one of the important thing is; you should know how to terminate a program because if you do not terminate a program well, then it will go on doing the calculations and you will never know whether it is finished or not, so therefore just as starting a program properly is important, terminating is also important.

Because once it terminates, if it does not terminate then it should give some message that something is going wrong, okay so terminating is also important, so these are the main ingredients.

(Refer Slide Time: 13:36)



Now, before we consider several other aspects, we have not talked much about commands in Linux, so, so far what we did; we uploaded a Linux operating system and in that started creating files by VI command, correct, then once you create the file, you save the file, then compile the file by gfortran file name and then execute, so the only commands we know so far are your VI command and gfortran command and ./a,out, okay.

There are now several other important commands in Linux which are very, very useful when you do programming, so in this slide I am listing all those useful commands and there are always help commands in Linux, so you can always get a help on that command. So, now let us see which are some of the most useful commands in Linux, so the first line says ls space –l, so this ls space -l list all the files in a directory, so this is a very useful command.

Because you want to know what are the files in a directory sometimes, your file may not be in the directory you are looking for so therefore, ls space - l, remember that it is not ls – l; ls space – l, it list all the files in a directory. If you just do ls, it will still give you a list, it will give a short list. When you say ls space –l, it will give you a long list that long list will tell you what is the size of the file and so on.

Then the other command which you will need all the time, you may want to create new directories because it is always good to have several directories and each directory will do a different job for you, so to create a directory, you will say mkdir new directory, so what you have done this with this command, is you have created a new directory with the name new dir and once you create a new directory, you may want to go for the present directory to the new directory, right, you want to change the directory now.

So, there is a structure, there are directories, there are subdirectory, subdirectories of sub directories, all Linux files have a structure like that so, when you create a new directory, so this new directory also will remain in the present directory and you may want to go from the current directory to this new directory, so that is how you see this command, cd new directory that means, you have change the directory from the present directory to new directory.

Now, if you want to go back to the original directory, cd.., so it will go back to earlier directory, so what are the commands we have seen so far; listing the files through ls command, creating a directory, changing a directory and going to the new directory and coming back to the new directory, so these are these 4 command. Then the other command you will need very often is copying files.

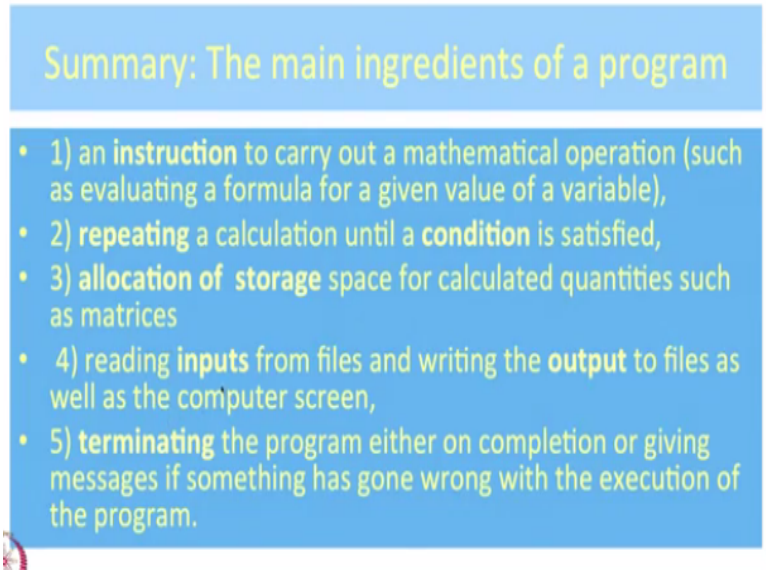
So, you may want to copy; make a copy of a current file, so the command is cp; cp is a copy command, copy file 1 file 2, what does this command do? It copies file 1 to file 2, so after this command, there will be 2 files; file 1 and file 2, okay. Now, suppose you want to remove this file, okay, suppose you want to remove any file; rm file 3, so what does these rm do? It deletes a file which is named file 3 but remember, be very, very careful when you delete a file.

Because once you delete a file, you will not be able to get it back, so always when you delete, be very, very careful, so it is good to have copies of all your files and save it on a hard disk suppose, you have done some work for several days, you have all those files and maybe your hard disk may crash after a few days, so whatever work you have done, make it a habit to save all those files on either a pen drive or some other hard disk, okay.

Again, to do that you can use a copy command to save from your hard disk to your external hard disk and finally, there is a help command, so when you say help, it will list all the commands that are of used in Fortran, okay and in many compilers there is also a man command, man is a manual, so if you say, man f77, it will give you all the features that are there in this command f77, man is a manual command, help is a help command.

So, as you go along learning, you will find that this help and man will be very, very useful and similarly, of course you can go the website, go to Google and see what are the Linux commands, what are the details and once you start doing computing for 2 or 3 weeks, all these commands will become part of your vocabulary and you will start using them, so it is not so difficult but you just need a little bit of practice.

(Refer Slide Time: 18:38)



So again, this summary once again, what were the earliest instruction; repeating, allocation of storage, reading inputs, terminating, this is a repeat of the last slide just to remind you what are all the main ingredients. So, now we will come up with a new topic, this is a very, very important topic because so far, what we have done, we have use several variables, what are the variables we used?

Ww as a variable and rtoww so, every variable took on 1 value, okay, so, so far whatever computing you have done, each variable takes 1 value but often we need variables which are dimension for example, many of you may be knowing what are matrices, what is a matrix? Matrix is an object, suppose I have a 10 x 10 matrix; a 10 x 10 matrix, there are 10 columns and rows, so this is a 10 x 10 matrix.

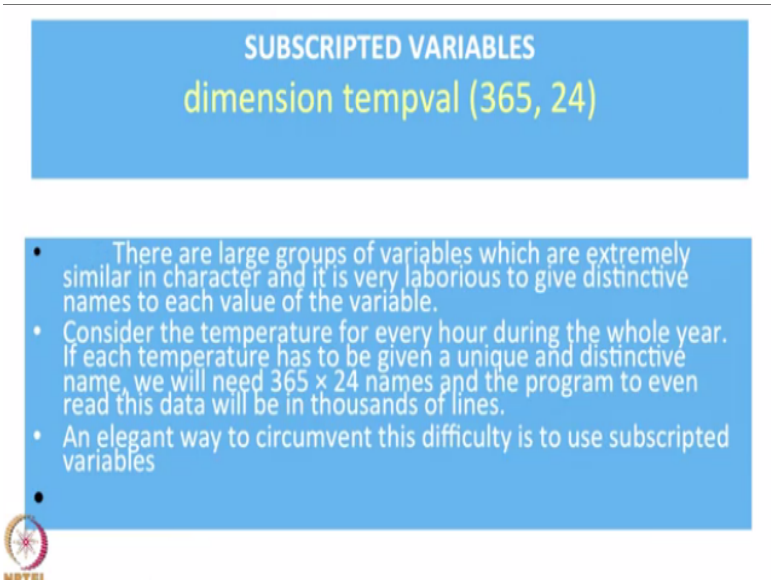
So, for a 10 x 10 matrix, there will be 100 objects associated with that matrix, so the question is you know the mathematics, you will use a, i , j, where i and j are subscripts but a computer does not understand subscripts because every line statement is given in one line, okay, so therefore you should have a mechanism to distinguish an object which is a normal variable like the root 2 or ww, you saw before or subscripted variables.

What are the subscripted variables? Like your matrix aij, aijk or there could be a say, vector, so what is a vector; vector, suppose the 3 dimensional vector, xyz or you can call it ai, i going from

1 to 3; a1, a2, a3, so this is a 1 dimensional variable with 3 components, a matrix is a 2 dimensional objects, a 2 dimensional objects will have rows and columns so similarly, you can have 3 dimensional objects, 4 dimensional objects.

Now, the reason for this is that you have often large data which has a similar meaning, so instead of giving so many names for each variable, it is good to give a subscripted variables for example, you know that in your class, you have roll numbers, why do we have roll numbers, why is that? Why did the roll number come in the first place or now, you have pan numbers, you have aadhar numbers, why are we having these numbers?

(Refer Slide Time: 21:07)



Because numbers are easier to rearrange and manipulate rather than names, okay so, so now the first example I will consider is a dimension variable which I am calling it so, I will call this a dimension variable, so I have call this dimensional variable, tempval 365, 24, this is the first dimensional variable we are using. The reason I am calling this dimensional variable tempval, there is a reason.

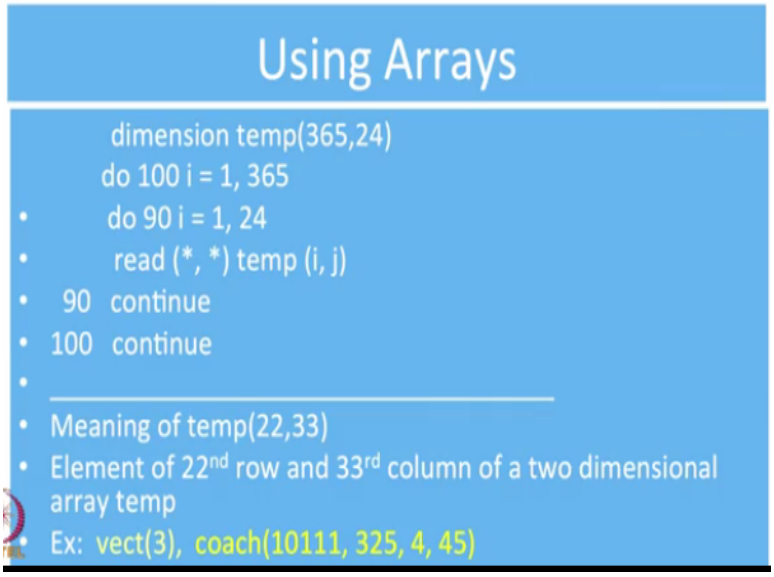
So, what we want to do; we are all chemistry students, so what we want to do; we want to know what is the average temperature of the atmosphere in the whole year, so this is my experiment now, chemistry experiment, it is not a great experiment in terms of new information but it is

useful also, it is useful, so I want to know the average temperature of my atmosphere in the whole year, so how will I do that?

One option is keep on measuring the temperature every minute and measure it throughout the year and get an average that will be too much because you know how much data you have to collect, so what I want to do; I want to collect data every hour for 24 hours, then every day of 365 days assume that it is not a leap year and I want to store that information in an array called tempval and this is now an array, it is not a simple variable like ww or root 2.

This is an array with 365 rows and 24 columns, okay, so this is an array, so this is the subscripted variable, so subscripted variables are dealt with differently than nonscript; non-subscripted variables, okay, so as I have describe the whole thing is written in the slide, you can look at it a leisure, I have created this subscripted variable content called tempval and how do I use it in a program that is our next task, okay.

(Refer Slide Time: 23:10)



So, this is not; this tempval takes 365 * 24 objects, it is not a single number, it is a very large object, so the way it is addressed is given here. So, this particular slide tells you how to use arrays, so the first thing you have to do if you are using an array is to declare the size of the array, you see that. First statement when you have an array, you declare its dimension, it says dimension temp 365, 24.

Earlier slide I had called it temp value, now I am just calling it temp because why do we give very, very names, temp will represent just 4 characters, so and temp also tells you it is the temperature, so remember give names to your variables which have some sense to you from your chemistry or physics point of you, so I have created an array temp which are 365 rows and 24 columns.

So, whenever I have such an array my read; so the way it is referred to in the program is given below, so now I want to, I have already made the measurements throughout the year, computed that is measured the temperature for 24 hours a day each hour once, 365 days, so this is my data, I have arrange the data into 365 rows and 24 columns, so the way I read it is in this manner, so my days go from 1 to 365.

So, one loop is there to read the row index, second loop is there to read the column index, now look at this statement, do 100 i going from 1 to 65, so what does our knowledge of do loop tell me? So, this variable i takes the value 1, does all the thing up to 100, comes back, takes the value of i = 2, go all the way to 100 and come back, this it will do 365 times, so when I am doing with i variable 365 times, I defined a second variable.

Now, this is not i, this should be j, so do 90 j going from 1 to 24 because it is a second variable, okay, so first variable is i, second variable is j, so what is the meaning of that? J goes from 1 to 24 up to 90, so the statements up to 90 are continued 24 times because j goes from 1 to 24, then once j does 1 to 24, it will go to 100 then i will increment, then again j goes from 1 to 24, i will be incremented.

So, these 2 do loops allow me to read those values of temperature for 365 days and 24 hours, so the way program will run, i will be 1 first time, j will be 1, read temperature 1, 1, first day first hour, then goes back, then j becomes 2, first day second hour, then j is 3, first day third hour, j is 4, first day fourth hour, until j is 24, it will read this that is first day 24, then it goes back to 100 continue, then it goes to this first line then, i = 2, second day.

Now, second day first hour, second day second hour like that second day up to 24 hours come back, third day; third day first hour up to third day 24 hours like that it will do for all the 365 days 24 times, then this whole part is over, so what is the advantage of this now? The advantage is through this set of 2 loops, I have read all my data and put in that variable, temp i, j. Now, what is; now let us see the meaning of this?

Suppose, I said temp 22, 33, what is the meaning of that? it is the 22nd day and 33rd hour but you know that there is no 33rd hour, so there will be a problem here, okay, see what was our array? 365 and 24, so when you have something like this, your row should be between 1 and 365 and column should be between 1 and 24, so if you give like this, there will be a huge problem because the computer does not know, what is the 33rd hour, the computer does not know.

Because you gave only 24 hours, if you want to read more than 24, your dimension should be more than 24, so this particular thing will give an error but if I have say 22, 2 that means it is 22nd day and second hour, so this is the meaning of this. Now, let us see before I conclude, I want to see what would you do suppose, you are in charge of the railway reservation system and you want to save the information about the reservations for all the trains for all the possible days.

Because now, the system you know that very well that when you reserve a ticket on your computer or PC, you are able to reserve sitting anywhere in the country, so how should the array be to save all the information about the reservation system that is what we will consider next time, so I will conclude this lecture now, so what we did this time is looked at some Linux commands, looked at some if statements.

There are many ways of writing if statements then finally, we initiated this time use of arrays, arrays are very, very crucial, so we will again continue use of arrays in the next class, so I will close here, thank you.