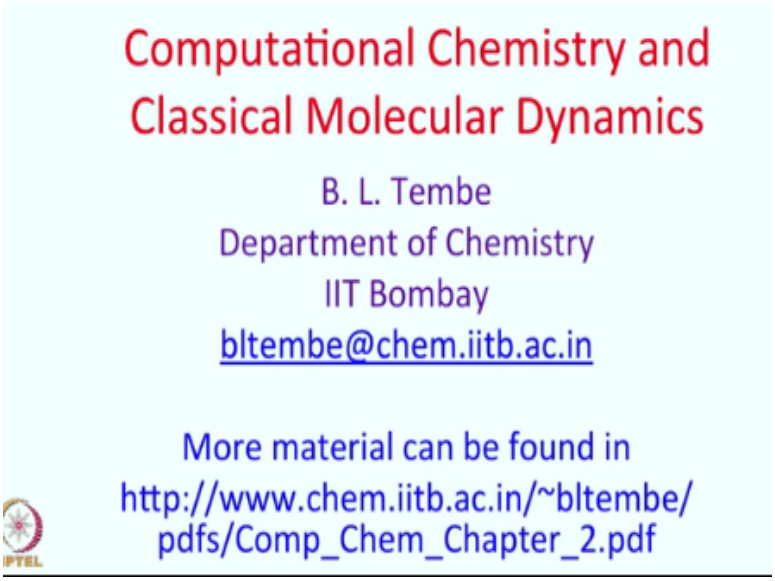
Computational Chemistry & Classical Molecular Dynamics

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

Lecture – 01
Introduction to Computational Chemistry

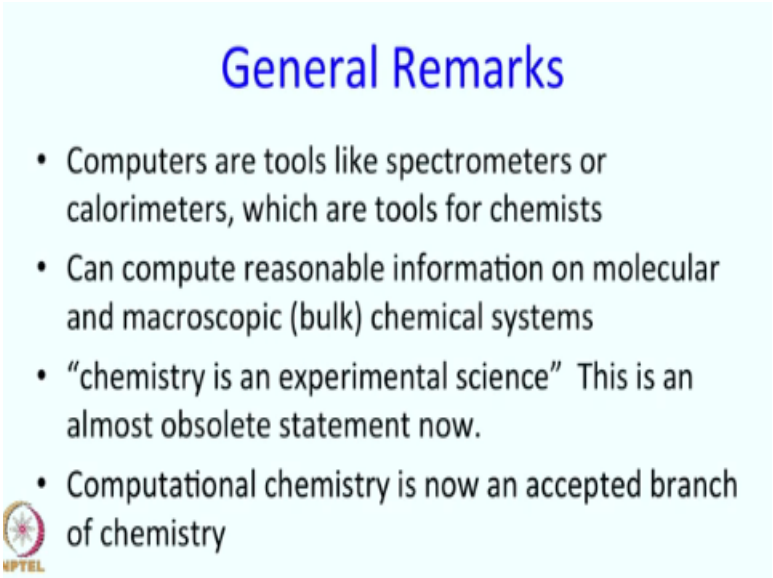
Hello and welcome to this course on computational chemistry and classical molecular dynamics. This is an introductory course which will introduce you to different aspects of computational chemistry, different software's as well as how to do classical molecular dynamics using public domain software's. My name is B. L. Tembe from the Department of Chemistry, IIT, Bombay.

(Refer Slide Time: 00:34)



My email address is bltembe@chem.iit.ac.in, most of this lecture material is already available on the website, what I will be doing is to introduce different concepts and give the details of all the new things that you will learn and give additional information as we go along. So, now the first question is; why are we interested in computational chemistry as far as chemistry is concerned, computers are tools just like spectrometers, calorimeters and other tools which are useful for chemists.

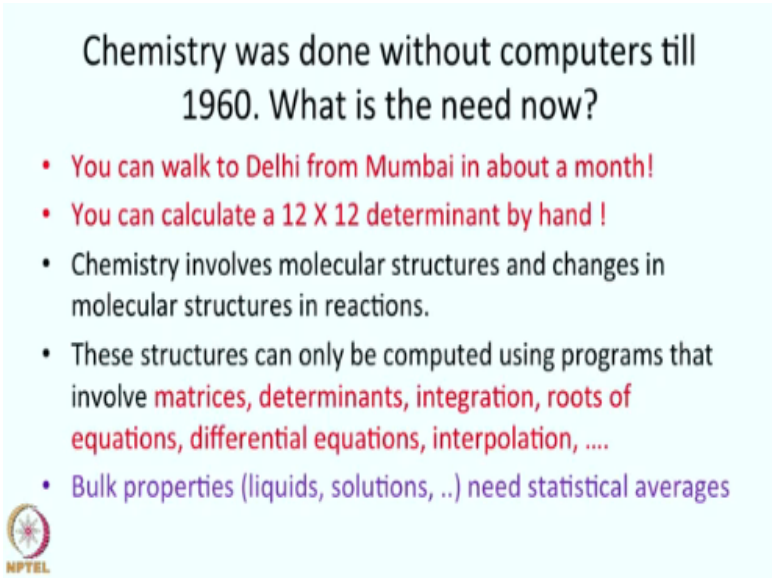
(Refer Slide Time: 01:24)



Now, using computers we can compute reasonable information on molecular and macroscopic chemical systems, we often hear in school that chemistry is an experimental science but this statement is almost obsolete now, because computational chemistry is now an accepted branch of chemistry just as your physical chemistry, organic chemistry and inorganic chemistry. Now, you may ask this question before 1960, we were doing lot of chemistry, when there were no computers at all.

Now, you may ask the question, why do we need computers now? Let us take 1 or 2 examples, suppose you want to go from Delhi to Mumbai, if you walk, it will take you about a month, so it is always possible to go from any point on earth to any other point by walking but it will take you such a long time that it is not worth the effort.

(Refer Slide Time: 02:14)

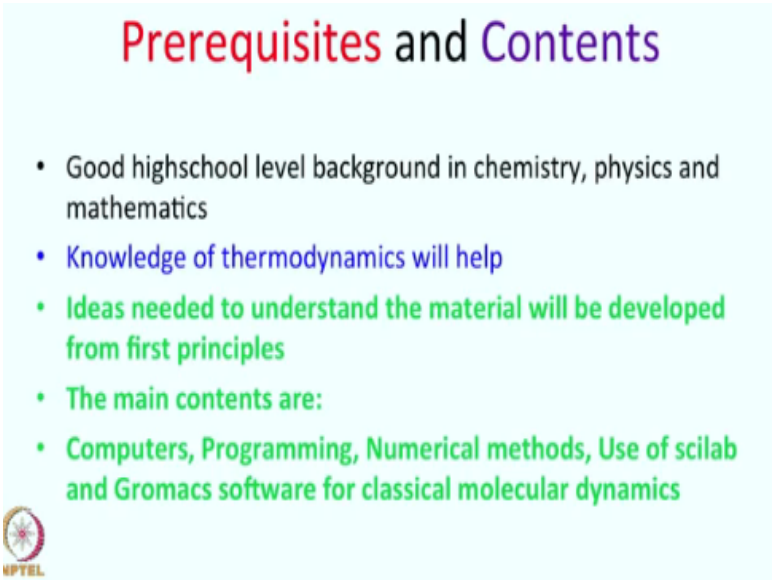


Another example would be many of you must have heard of what the matrix and a determinant is; for example, consider a 2 x 2 determinant, a 2 x 2 determinant would be suppose, the matrix is ABCD, the determinants AD – BC, so calculating the determinant of a 2 x 2 matrix will take just about 30 seconds, now consider a 12 x 12 determinant to calculate the determinant by hand, it will take you more than the time of your life.

So, if an object takes more than a lifetime to calculate certainly, you will use any technique or instrument that will help you to do this faster, so one of the main advantage of computers is that in a very, very short time, it will give you a lot of information about molecular structures and changes in these molecular structures during reactions and after all that is what chemistry is all about. So, these structures can only be computed now using programs that involve matrices, determinants, integration, roots of equation, differential equation, interpolations and so on.

And bulk properties like entropy, energy, enthalpy these all can be calculated as statistical averages or properties of molecular systems, so the final goal of this course would be how I can compute things of interest in chemistry using techniques like statistical averaging. So, let us now go to what are the prerequisites for this course and what are the contents.

(Refer Slide Time: 03:51)

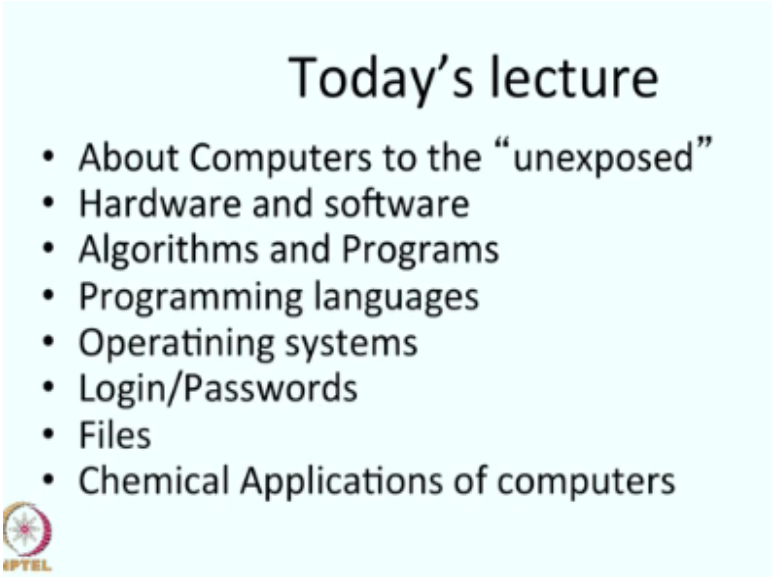


So, as far as prerequisites are concerned if you have a good high school background of chemistry, physics and mathematics that is more than sufficient to learn all the contents of this course and of course, you will need some knowledge of thermodynamics that you have already studied in your 11th and 12th standards, whatever ideas that are needed to understand the material this course will be developed from first principles.

As I mentioned earlier, the main contents are computers, programming, numerical methods, use of scilab and Gromacs software for classical molecular dynamics. So, what are we going to do in today's lectures? Today's lecture is going to be a very elementary introduction about computers to the unexposed nowadays, everybody knows about computers right from the school level.

But when we studied chemistry, we had not seen computers till we finished our master's degree, so we will talk about computers to the unexposed.

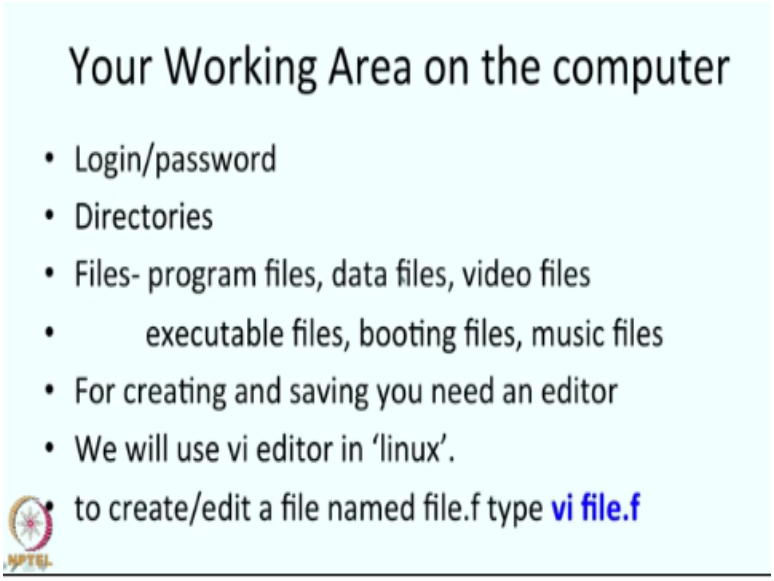
(Refer Slide Time: 04:44)



We will talk of hardware and software, algorithms and programs, programming languages, operating systems, login, passwords, files and finally, we will also discuss chemical applications of computers. So, whenever you are using a computer, there is a part of the computer which you are allowed to use through a login and passwords, so it is like just as you have a house where you can live in a city, even on a computer there is an area of that computer which you kind of own and change at will.

So for that you need a login and password, so that you can work in that area and in that area, you can create several directories; directories are like various cabins where you can store information.

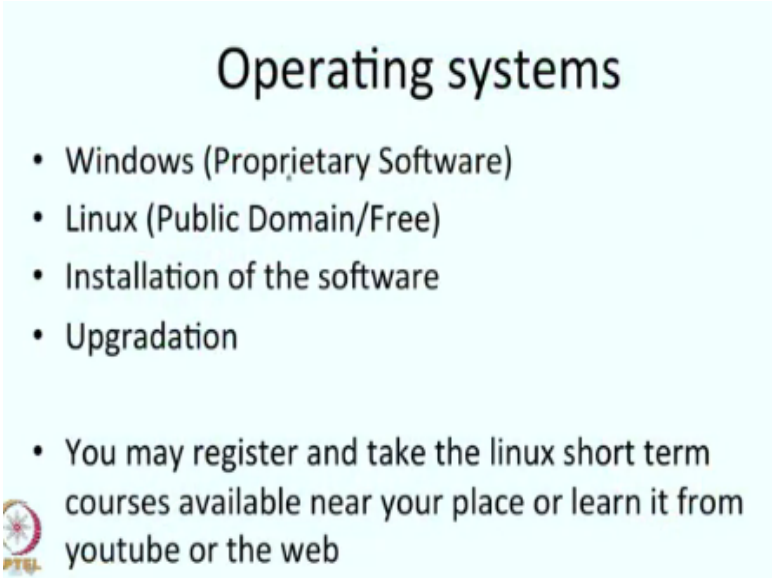
(Refer Slide Time: 05:27)



And the files that you will be using every object in a computer is a file for example, you have a program file, data file, video file, executable file, booting files, music files, video files so on, these are all files which you can save and alter the way you want okay and the operating system we will be used for our course is Linux, now why we use Linux? Because it is free software, you can download it from the internet, there is no cost.

So to everything that you do will be through the creation of a file, so to create or edit a file suppose, the file name is file.f, this is the name of that file, so to create or edit that file I have to just type vi file.f in your Linux operating system, so our work will start with creating or editing files, so I have called it file.f. Now, .f is an extension of the file, so different files will have different extension were interested in .f files.

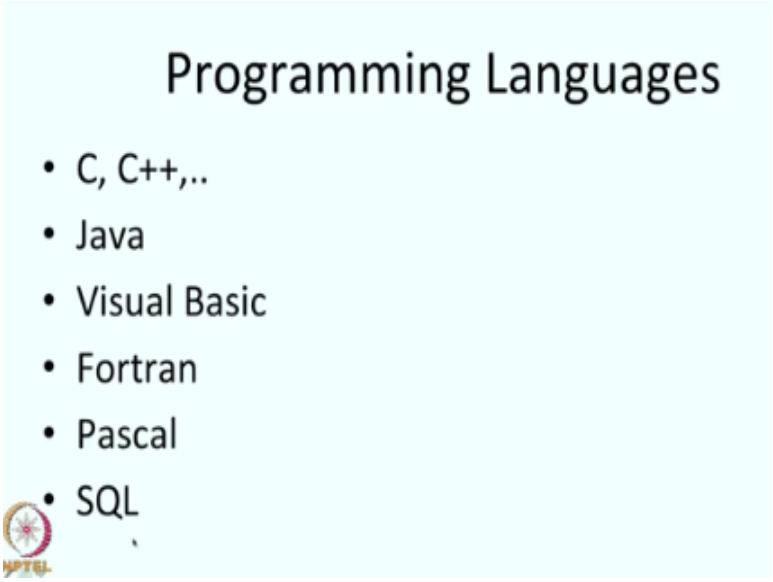
(Refer Slide Time: 06:36)



Because we will be mostly working in for time to begin with, you must be all very familiar with windows software that is a proprietary software, you have to pay for it and as I mentioned, Linux; it is a public domain software, it is completely free and you can install the software on your own from a website and you can upgrade it as you want a newer and newer version of it. So, if you are not completely familiar, you can register and take a Linux short term course available near your place or you can learn it from YouTube or the website.

Whatever you need for our work, we will discuss it as we go along in our course, so you must have all heard about computer programs. Computer programs all will need a computing language just as different states have different languages, computers also have different languages.

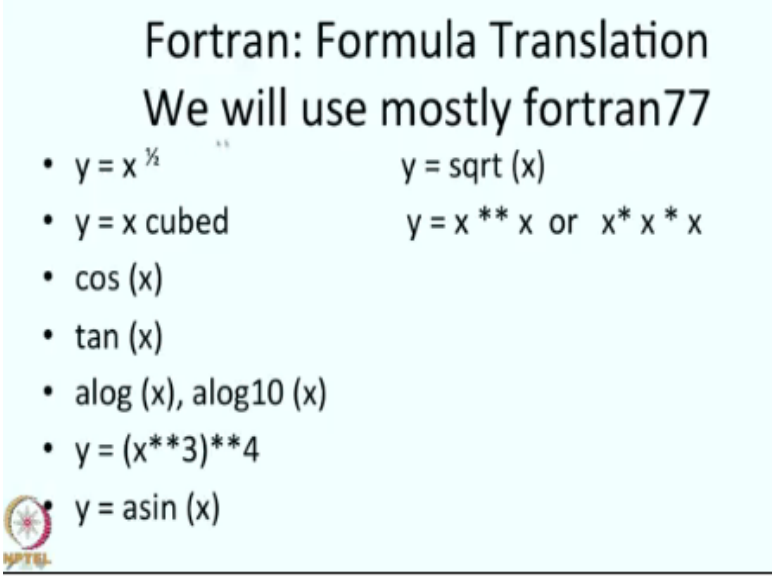
(Refer Slide Time: 07:26)



The popular ones today are C, C++, Java, Visual Basic, Fortran, Pascal, SQL, there are many languages, they keep changing in fact, there are many languages I learnt and which are absolute now, so as we go along even the language may change but the basic logic and ideas are very uniform, concepts are the same the way you write in different language it is different.

Mostly, we will use Fortran because it is an old language and many of the chemistry programs are still written in Fortran and there are programs now to convert from one language to another, so if you write a program in Fortran, you can always convert that from Fortran to C, if you have program in C, you can convert that from C to Fortran, so that option is there but once we understand different concepts of this language, then we will be able to do whatever programs we want.

(Refer Slide Time: 08:23)



So, now let us see what Fortran is all about; Fortran is nothing but formula translation, so we will be using mostly fortran77, now what is the formula translation? In mathematics, you will write y = x to the power of 1/2 but when you do a program, the entire line in a program should be in one line, you cannot have superscript and subscript and so on, so in the program this mathematical expression is written as y = square root of x.

See that this is entirely in one line now similarly, suppose you want y = x cubed that superscript 3, I cannot write on one line, so the way it is written on a computer or the way it is translated in Fortran is y = x * * x, this * * means raising an object to the power. In this case, y is raised, x is raised to the power of x, the other way to do it would be x * x * x, okay, so there are certain differences between x to the power of x and x cubed, so we will come to it at a later stage.

Then, the other functions that you are used to or something like y = cos x, this is a function, tan x, this is another function log x, so actually in Fortran, you do not call it log x, you call it alog x because this is the log function, so this function is to the base e, whereas alog 10 x is to the base 10, most of the logarithm that you use are to the base 10, so alog 10 x will be a function where you have calculated the log to the base 10.

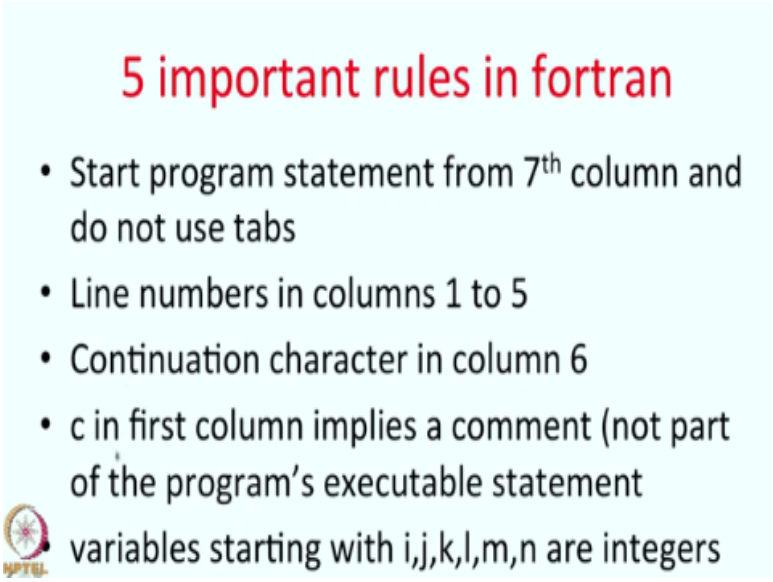
So, now you consider another function y = x * * 3, the whole thing * * 4, what do you think this is? This is nothing but inside the bracket, you have x raised to power 3 and the whole thing is raised to the power 4, so this is how you write a complicated function in just the one linear way on a single line that you type on your computer okay. So, another function be suppose, you

want a sin x, y will be; a sin x will be your hyperbolic sin or in some computers, say it could be sin hyperbolic x.

So, different functions have different representation in your programming language, so as you go along, you will be able to do all those transformations. Now, let us see what are some of the important rules in Fortran? As every language has its own character, Fortran also has its own character for example, if you write Hindi or Sanskrit, you start writing from left to right but if you write Urdu, you will be starting from right to left.

So, each language will have its own rules and when you write your programs in Fortran, all the rules you should be very familiar with okay.

(Refer Slide Time: 11:28)



So, on this particular slide, I am listing some of the most crucial rules that are required when you write the program, so what is the first rule? Every statement that you write in a program should start from the 7th column that is you will not start writing from the 1st column of your line, you start from the 7th column because 1st column to 6th column have a different role for example, you see the next line; from 1st to 5th column, they are used to give line numbers to your lines.

Usually, you do not have to give any line number but the first 5 columns are used to give line numbers to the program and now suppose, your line extends more than 80 or 100 characters, if the line is very long, you cannot write on a single line, so you will have to go to the next line, so

what you do; you start typing from the 7th column on line 1 and it extends more than the 1st line, in the 2nd line you give some character on column 6.

If you give some character on column 6 for example, say 1, 2, 3, it could be any character, if you give a character on column 6, it says that the earlier line is continued on the 2nd line, so suppose your material is not complete even in the 2nd line, you may want to go to the 3rd line then, what you do; go complete your 2nd line, go to the 3rd line and in the 6th column of the 3rd line, put something 1, 2, 3 in the 6th column and continue.

So that way you can write long programming lines using Fortran and many times, when you write a program, you may want to write some comments because when you write a program today and look at it say after 6 months, you will not know what the program is all about so, you may want to write several comments, so that it will help you to recognize what you have done, these are like notes that you write.

So, any comment line will start with a C in the first column, so you have a; you have a line and if you put a C in the first column that is just a comment and the program will not execute it. So, these comment lines are very useful, so using these comment lines you can find out what the program is all about or what your intention was or you may have taken the program from some reference, you can give that reference in that comment line.

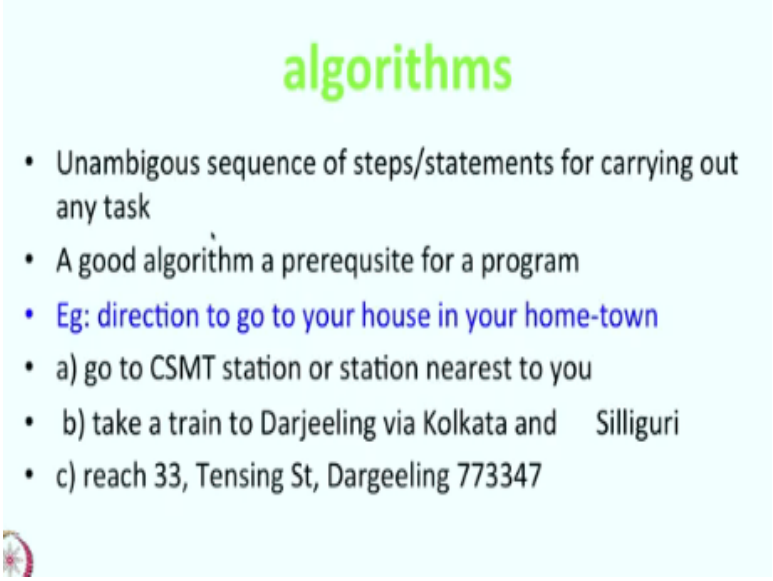
And you can do many things using comment cards in fact, writing good comments about your program is very useful in the future, then the other most important thing which you will need is that each variable will be having a name in the program just as you have your name and my name and in the computer program, you have x y, z w, so many constants are there in chemistry you have, Planck's constant, Avogadro number, each variable has to have a name.

So, the name has to begin with one of the 26 letters of the alphabet, what Fortran does; computers distinguish between integers and real numbers, integers are 1, 2, 3, 4, 5, 6, whereas real numbers are 1.23, 1.44, so on a computer representing integers is different from representing real numbers, so what Fortran does if you have integers, those integers will have variables which start with letters i, j, k, l, m and n.

So, what you have to do; whenever you have variables in Fortran, there will be 2 types of variables; real variables and integer variables. All the integer variables you start with i, j, k, l, m, n, so this looks like a very complicated thing when you begin with but as you start programming, this will become your second nature, so it is not such a complex thing, so as you go along you will be able to do this, okay.

So, now before you write a program, the entire idea has to be put in what is called an algorithm, so you would see; you look up the meaning of the word algorithm in a dictionary.

(Refer Slide Time: 15:40)



So an algorithm is an unambiguous sequence of steps or statements required to carry out the task, so if you want to write a good program, you should have a good algorithm based on which you are writing the program, okay. So, let us consider an example suppose, you want to go to your house in your hometown and somebody will ask you to give directions, so if you give directions, the direction should be given in such a way that the person will reach your house without asking anybody.

If you just give him the direction, if you just give him the home address and ask him to go, he will not know where to go. So, for example, let us see the following algorithm which gives you how to reach your address suppose, your address is 33, Tensing street, Darjeeling 773347, this is your address, now you have a stranger and you want to tell him how to go to this address, so what will you do?

You will say that let him go to the nearest railway station, next to you of course, you have to do again direction how to reach that railway station but we will come to that later then, he has to take a train to Darjeeling and how he will go to Darjeeling; he has to go to Calcutta first and then from Calcutta to Silliguri and then from Silliguri to Darjeeling and once he reaches Darjeeling, he has to find that address.

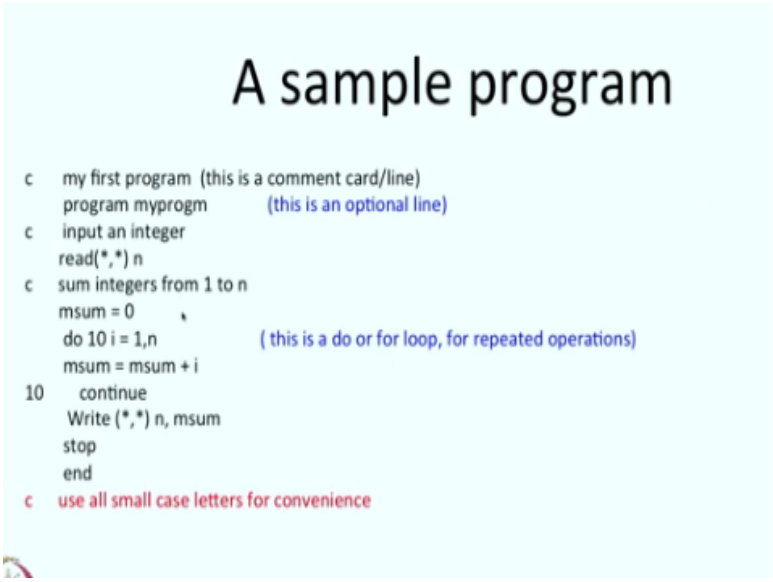
Nowadays, you have Google and Google Maps, so before you tell anybody to go somewhere, it is best to show on a Google map what that address is because once the person knows the map, he need not talk to anybody because the map itself gives you an algorithm after all what is a map; map is just all the objects are there, they are linked through roads, so the moment you see a map you will know all the roads that connect to the house.

And he himself can come with an algorithm. What is his algorithm; he will draw a line from his present location to that location in Darjeeling, so that is how he will reach that place, so your algorithm is really follow the path given by the line and each line will tell you whether to turn left, right, very unambiguously there is; so one of the things is in an algorithm everything has to be very precise

For example, suppose your mother tells you buy some vegetables in the market, you will go to the market then there are so many vegetables you are confused as to what to buy, then you will again call her so that is not a good algorithm. A good algorithm will be your mother will tell you half a kilo of this, quarter kilo of this, 100 grams of this everything precise, then that is an algorithm.

So, a good algorithm is a basis on which good programs can be written but once you get more experience with this, you do not really need algorithms; you will start writing programs from the scratch. So, what I will do; now, I am going to give an example of this program, so let us see an example of this program, then you will be able to proceed in this way.

(Refer Slide Time: 18:40)



So, what we have done is; this program, it is a very simple program, it will read some integers and it will sum those integers and it will write the result on your screen, so this is a simple program which will read some objects, which will sum those objects and write the result on the screen. So, let us see how this program is written, so the first line it begins with a comment, first character, first column when there is a comment, we already said it is a comment card, it will not execute, it will not do anything, it is just information.

So, this says this is my first program, so this is a comment line, so the next line is program, my program, this is an optional line, you do not really have to say it but it is always good to give a name to the program, so what I have done, I have called it my program and see that I have started this program on the 7th column, remember what was our basic rule that everything used right as a program line should start with a 7th column.

So, I have; this is my 2nd line, the 3rd line c; it says input and integer, so when you see this line you will know that what you want to try to do in your program is that the program should read an integer, now next line really is your first important line in a program, it says again from the 7th column, read into bracket *, * bracket complete n. So, remember I said earlier that whenever a variable begins with i, j, k, l, m, n, it is an integer.

So, what this line says, read an integer n from your screen, so how do I tell the program it is from the screen that is what this into bracket * , * is doing whenever, you have this into bracket * , * you are reading from the screen here and you are writing to the screen, so in the next line

you have read that integer n from the screen, so once you read that n now, again I have a comment card, the comment card says that some integers from 1 to n.

So what your program is going to do; your program is going to sum 1, 2, 3, 4 all the way up to n, n could be anything of course, n cannot be infinity because if n is infinity, your computer will not be able to do it, n is some integer typically in 1000, it can be 10,000 okay and what you want to do; you want to sum all those numbers up to n. Now, suppose you have to do this by hand, I ask you to some 1 to n, so it will take you forever unless you know, what is the formula for that sum?

In fact, it was Gauss who gave first the formula how to sum from 1 to n but that we will discuss later so now, we want to do it through a program, so up to this point you have read what is that integer n and you want to start summing whenever, you start summing numbers you want to initialize sum to 0, okay because when you want to add 100 numbers, you go on taking the first number to that number, you add the second to that number you add the third.

So, there is some box which you initialize and keep on adding the sum into that box, so what I have called in the next line msum = 0 that says that I have a variable called msum, I have assigned the value 0, in any program equal to is not really mathematical equal to, it means that whatever is on the right side that object is assigned to the variable on the left, msum here is a variable, to that variable I have assigned the value 0.

So, now my real programming starts, I want to start with 0 and go on adding first the number 1 to 0, so my sum is 1, then I will add 2 to that my sum is 3, then I will add 3 to that my sum is 6, so I have to keep on adding 0, 1, 2, 3, 4 all that into that box, so this whole process of repeating something is called a do loop, it is an iterative procedure, an iterative procedure is a procedure where you keep on doing it repeatedly until some condition is satisfied.

For example, I tell you keep on counting from 1 to 30,000, until you reach 30,000, so what you will do; you will keep on counting 1, 2, 3, 4 until that 30,000, you will keep on counting, so that is your repeated operation. Now, what I have told in the program; in the program I say, do 10, i going from 1 to n, so this is a statement which tells the program that this is a repeated operation you have to do until certain condition is satisfied.

So, the repeated operation will come after the do statements, so do 10; this 10 is not a number 10, it is a line number, what it tells the program is; up to the statement which has a line number 10, all those operations are repeated, okay. What are the operations? You start the operation with i going from 1 up to n okay, so this means all the statements up to line number 10 are repeated starting with i = 1, then up to i = n, so the first time i will be 1.

So, I go to the next line, the next line says msum = msum + i, so you know that already the m sum was defined to be 0, so what this line does; now go to the right side msum had the value 0, to that value of msum, it has the value i. What was the value of i? i was = 1, the first time, so the next value of msum is 0 + 1, so msum will be 1 and once it does that it goes to the next line it says 10 continue.

So, 10 continue means, go back to this do statement, so when I come to this second time, i has now become = 2 now, because first time it was 1, next time it will be 2, so when i = 2, now I go to the next line, again it says msum = msum + 1, now msum was already 1 in the last calculation to that msum, i had 2, so the sum is 3, so the new value of msum is 3 and it says continue, so continue means go back, it goes back to this do statement.

So, when it goes back to the do statement now, I has already completed the work for 1 and 2, this time n will be 3, so when you go to the next line, msum will be old value of msum + this 3, so already there was 3 before, to that I add a new value 3, so my msum will be 6 and I go to the next line I continue, next time when I do that now, n = 4, my msum is already 6, to that I add 4, so it will be 10, new value of msum, it becomes 10 and it goes to continue.

So, I keep on repeating this operation until I calculate up to n, suppose n is 100, then I will be doing this loop all the way from 1 to 100 and once n is 100, when his goes to 10 continue, it knows that I has already reached the maximum value of n, so it will come outside the loop, so this do loop has already a conditional statement that do all these operations up to n = the maximum value which is assigned in the read statement and it comes out.

Once it comes out write *, * n, msum, now n is the starting value of n you have write and it will print that n and it will also write the value of msum, so once it has done all the calculation, you tell the computer to stop and end the program, so again I have a comment here, I have said use

all small case letters for convenience because many computers do distinguish between capital letters and small letters.

It is best that you distinguish capital from small, so that there will be no confusion at the end, it may so happen that it may take small and capital to be the same but since you are a programmer everything you do should be unique, so this is the simplest program you started with, so what we will do next time; we will again start with this elementary program, discuss what were the rules we had and what the program is then we will see what to do next with this program.

Because this program is in some computing language, now I was to convert that into a situation which is convenient for me. We will stop here and in the next class, we will continue at this point. Thank you.