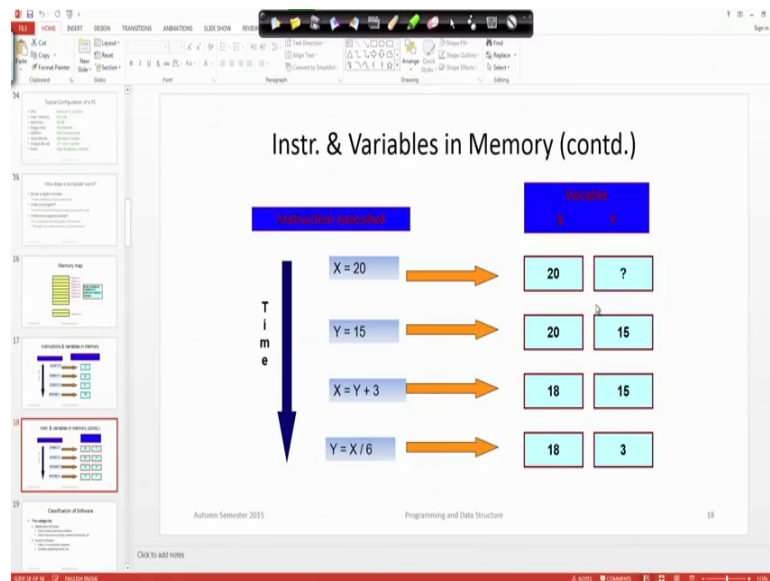


Problem Solving through Programming In C
Prof. Anupam Basu
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture - 06
Types of Software and Compilers

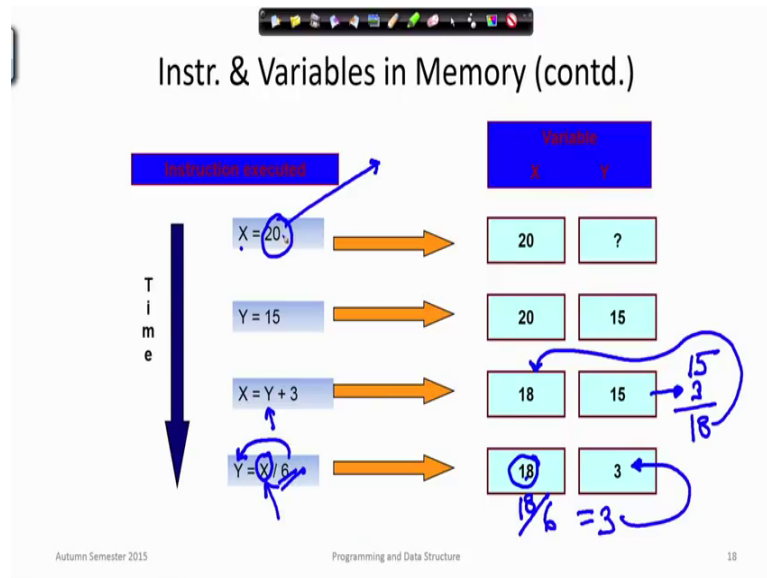
In the last class, last lecture, we were discussed about variables and values

(Refer Slide Time: 00:24)



And we explained that variables are mapped by the compiler to different memory locations. So, whenever we mention about any variable X Y or Z, each of them corresponds to a memory location, each of them corresponds to a memory location and each of them corresponds to a memory location

(Refer Slide Time: 00:48)



And whenever we assign some value like 20 to X; that means, in a particular memory location that value is written. So, we had done this example once again, we quickly go through it that X is being assigned 20. So, the memory location corresponding to X is getting the value 20 whereas, the memory location corresponding to Y can be anything, when we assign in this statement the value 15 to Y.

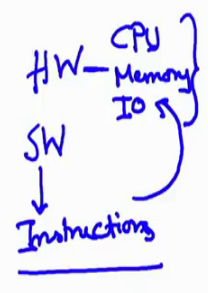
Then 15 is written in the location, corresponding to Y, when we do Y plus 3 and assign it to X; that means, actually we are reading this value of Y, here reading the value of Y which is 15 and we are adding 3 with that and we are getting 18 and that 18 is written into this location, all right. This may appear to be very simple, but this way of thinking or way of looking at the things will come in very handy as we will see at a later, during the later phases of programming.

Now, here again when we are, what is meant by this statement? That the value how will I read? I will read it like this that the value of X value stored in X that is 18 will be divided by 6 and that result will be stored in Y all right. So, X was 18, 18 has been divided by 6 and we get 3 and that 3 is written in Y. Now, here there are couple of things, that what is this 18? What is this 20? These are values and whereas, these are variables, these are also known as constants. Constants are the values which do not change during the execution of the program, next we can now, we can think of the software.

(Refer Slide Time: 03:26)

Classification of Software

- Two categories:
 - Application Software
 - Used to solve a particular problem.
 - Editor, financial accounting, weather forecasting, etc.
 - System Software
 - Helps in running other programs.
 - Compiler, operating system, etc.



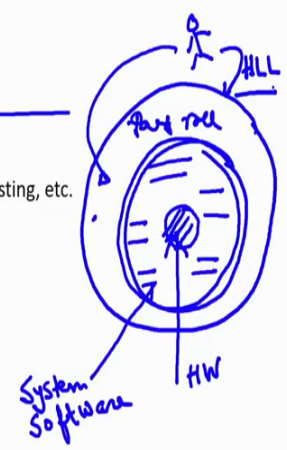
Autumn Semester 2015 Programming and Data Structure

Now, you know any computer system; consist of hardware as well as software right. So, we will have some hardware as well as software. Now, the hardware is consisting of the C P U, the memory, the I O devices, all those things are hardware and the software is the instructions, that this hardware that is executed by this hardware, the instructions that are executed by this hardware constitute the software. Now, software can be of two types for example, first one is the application software; application software is the software that we write.

(Refer Slide Time: 04:16)

Classification of Software

- Two categories:
 - Application Software
 - Used to solve a particular problem.
 - Editor, financial accounting, weather forecasting, etc.
 - System Software
 - Helps in running other programs.
 - Compiler, operating system, etc.



Autumn Semester 2015 Programming and Data Structure 19

So, we can just have an idea of this, through this onion type of diagram at the core, we have got the hardware. The hardware is here and I am putting two layers around this and the user is standing somewhere here and the user is not directly interacting with the hardware, because the hardware nearly understands ones and zeros and it is very difficult for the user to write in ones and zeros. So, the user will write in some high level language H L L, in which the user writes and the system automatically converts it into a way that is understood by the hardware.

And what is that automatic way of converting it, we have seen that is compiler, is a software compiler is again a software right, a compiler is software which converts before coming to application software. Let me talk about system software. So, we know that when the user has written something in high level language, that is converted by a program called compiler into the machine level language or high machine level language right, which the hardware understands.

So, the compiler is let us mark it like this. These are part of this layer, which is the system software similarly; operating system is another very important software, that is lying in this layer, internal layer which enables the user to use the computer in a much more user friendly way and in a much more efficient way. So, operating system compiler etcetera are the core, very important elements of the entire computer system, without which we cannot, we would not be able to use the computer in, as efficient way as we do it now a days.

Now, given this hardware and this layer of operating system, and compiler, and other system software, now, we are in a position to write some programs for our day to day use, for example, a company wants to find out the salary information of the people, they can use some pay roll software, here or for example, you want to design some data analysis software that will take some data and using a particular software will analyze the data statistically and give you some good insights.

So, all those things, the user is writing and they are forming the application software. So, most of the time the programmers, who are not systems programmers, not the system designers, but just users they mostly use the application software given this, we come to a very important software.

(Refer Slide Time: 08:10)

Operating Systems

- Makes the computer easy to use.
 - Basically the computer is very difficult to use.
 - Understands only machine language.
- Operating systems make computers easy to use.
- Categories of operating systems:
 - Single user
 - Multi user
 - Time sharing
 - Multitasking
 - Real time

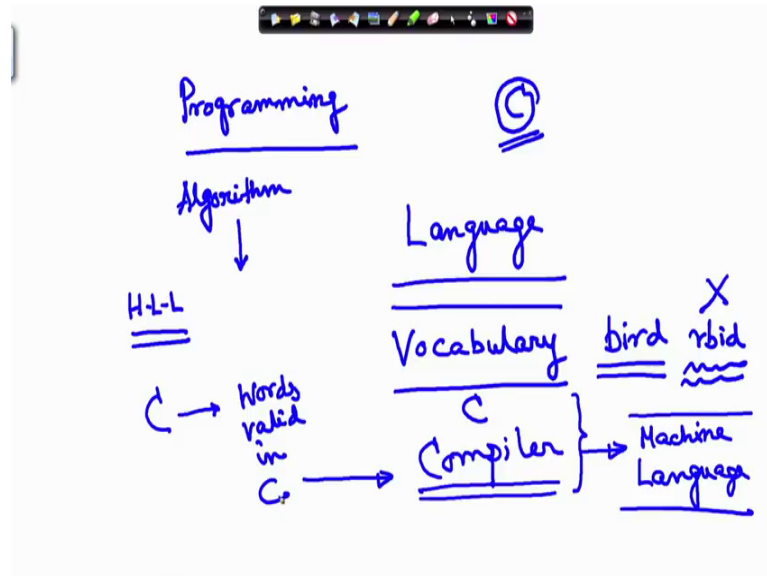
Hand-drawn diagram: A central circle labeled 'Compiler' has an arrow pointing to a larger circle containing 'Linux/Unix' and 'Mac OS'. This larger circle is enclosed by a bracketed area labeled 'Windows'.

Autumn Semester 2015 Programming and Data Structure 20

We just now mentioned, that is an operating system. Now, you know that you are averred of typical operating systems like windows, Linux or Unix, now a day's Apple is becoming popular Mac OS, all these things are operating systems.

Now, what is the operating system? The operating system is a layer around the hardware which enables a user to use the software, use this computer system. I am sorry, use the system in a much more friendly way. Now, there are different varieties of operating system; single user operating system, multi user operating system, etcetera. Now, this operating system also activates another system software that we have talked about, that is a compiler right. The operating system will call or will activate the software system, software call the compiler, when you want to run a high level program, given this background. Let us now move to discussion on programming.

(Refer Slide Time: 09:41)



We know by now that programming means we have to express our intention of solving a problem by executing a number of steps and those number of steps, once again you know that by now, we start with the algorithm and that algorithm can be expressed in different ways like pseudo code or flow chart and then the programmer actually writes them in a high level language, some high level language.

Now, what we are going to discuss now, is a particular high level language, which is called C. We are taking C just as an example of an high level language, because we have to express the logic in the form of some high level language. We are taking C as an example and as I have mentioned earlier that the logic, the style and the philosophy, remains more or less the same across different programming languages like Java, C plus plus and others of course, C is the simplest to start with.

Let us see now, we are using the term language, think of a human language; any language is constituted of some vocabulary right. The vocabulary say for example, in English, the vocabulary consists of some words right. Different words like a bird is a valid word in a, in the English vocabulary now, but if I had written r b i d that is possibly not a valid word in the English language vocabulary all right.

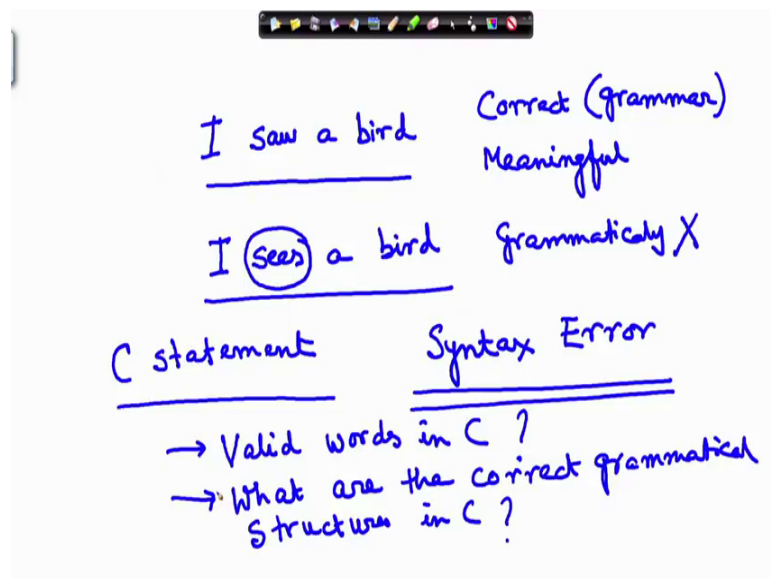
Now, we have got some valid words. Similarly, C will also have some valid words, which will see through, which we can express the basic elements of a C program, just as English sentence is built using English, valid English words, otherwise the meaning will

not be understood in the case of C programming language. Say C will have, it is own vocabulary right. The words valid in C now, if I had written some word in English, that is now some string, some patter in English like r b i d that unless this has got some special meaning, this will not be an, if this is a part of a sentence, this will not be very clearly understood by anybody.

Now, when I have written a C program, who is going to understand this, for whom I am, I writing this C program, I am writing this C program, for the compiler C, compiler all right. I am writing it for the C compiler and the C compiler is responsible to understand this and just as we understand an English sentence open the door. So, we understand the meaning of that sentence, we go and execute that we open the door.

Similarly, in C if we write something, unless the compiler understands this, it will not be able to convert it to the machine language, which will be executed by the hardware or the computer right. So, the C program must constitute of valid C words and we will see what are the valid words and what are the rules for that. Now, the next thing, if I write a particular sentence

(Refer Slide Time: 14:38)



I saw a bird that is a valid English sentence, why is it valid? Because it is grammatically correct and also it is carrying a very clear meaning. It is a meaningful sentence right, this is both correct, grammatically by grammar and also it is meaningful. Now suppose, I wrote it, wrote something like I sees a bird. Now, this is grammatically wrong; however,

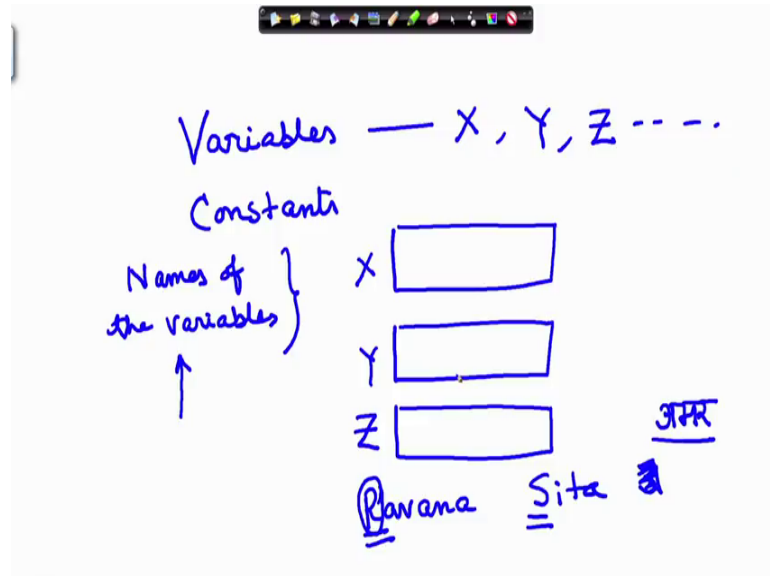
it conveys some meaning, I understand that the person who wrote this is weak in grammar, he is not very much conversant with subject, verb, agreement, but still I can make a meaning out of it.

On the other hand for a compiler, if I write a C sentence, let me call it not sentence, let me call it C statement. Now, a C statement; that means, a statement in the language C will consist of some valid words in C and also it will have to follow some grammatical rules of C. Unfortunately, here although it is grammatically wrong, I could understand the meaning of this, but a C statement, if it is grammatically wrong, grammatically according to the grammar of C, if it is not in tune with the grammar of C.

Then is grammatically wrong and since this C sentence will be interpreted not by a human being, but by a machine, computer hardware smart, it may look like is basically, not as intelligent as human beings. So, that as of now, whatever we can make out the meaning of it a compiler a C program will not the compiler, will not be able to make out correspondingly the correct machine language cannot be generated therefore, if there is something that is grammatically wrong according to C grammar that will be indicated marked by the compiler as a syntax error all right. So, unless we write something in the correct syntax, there is always a chance of it will it not there is always a chance it will; obviously, lead to a syntax error and the compiler will not produce the corresponding machine code, all right.

So, we are now supposed to learn what is therefore, we need to know what are the valid words in C and what are the correct grammatical structures in C. Now, if we learn both these, we learn the C language similarly, if you want to learn any other language computer language you have to exactly know this things what are the valid words, in that language and what is the correct grammatical structure in that language. So, given this we will start looking at C programs, even before that we once again recapitulate.

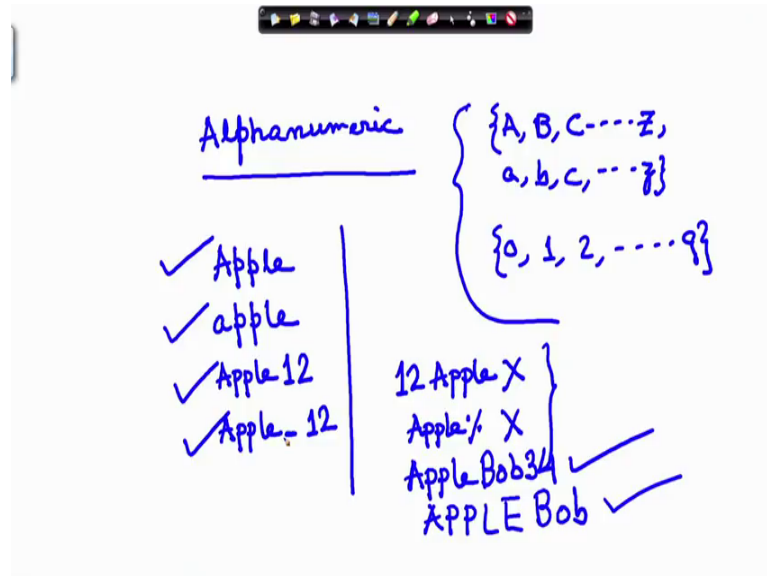
(Refer Slide Time: 19:34)



Whatever we are expressing them using some variables constants, right and the variables I have got, we have shown X Y Z, etcetera, at variables. So, each of this variables which are nothing, but memory locations are been given some names right. So, we have to give some names of the variables all right. There are some rules for naming them just as in English, we start a proper noun with a capital letter Ravana, all right Sita, we write them with the capital letter this is the rule of English; however, if you write in any Indian language, say you write anything, Amar you write that there is no question of any capital letter here, but it is a, it is a property of English, it is a rule of English that the capital first letter of a proper noun must start with the capital letter.

Similarly, for naming the variables in C there are some rules. So, here we have seen X Y Z, etcetera. These are the variables. So, there are some rules for naming the variables, we will come back to this again, but first of all any string of alphabets say, let me introduce one word; alphanumeric.

(Refer Slide Time: 21:45)



Alphanumeric means what alphabets and what are the English alphabets A B C up to Z small letter a b c up to z that is alphabet and numerals, we know 0 to 9 all right 0 1 2 3 up to 9. Now, an alphanumeric means A union of both these either alphabet or numeral. So, a variable name in C can consist of any alphanumeric character now.

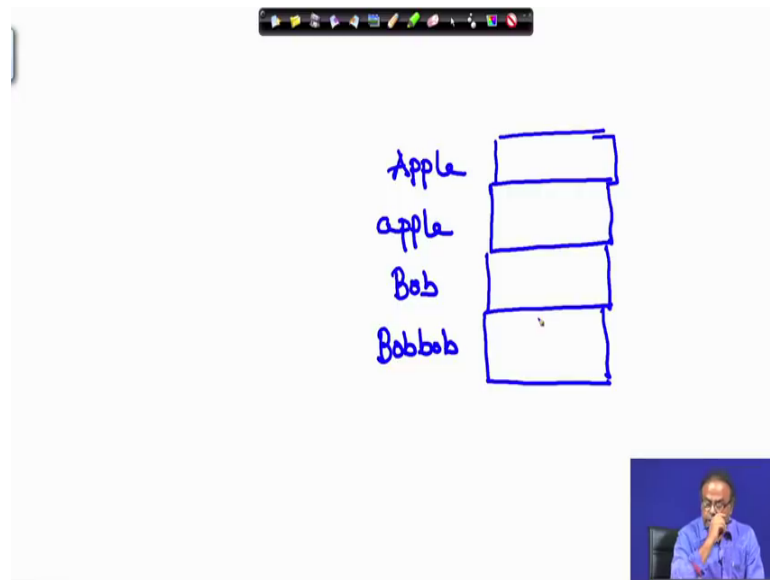
So, it can be say, Apple is a valid variable name in C, again if I start with small a this is also valid variable name in c, but although these two are same, since I have put in different characters, one capital, one small, these two will be treated as two separate variable names all right. Similarly, I can write, say Apple 1 2 that is also valid variable name. Now, there are some special characters like underscore, that is allowed like I could have written something like this, Apple underscore 1 2 that is also valid variable name.

However I cannot start a variable name with a digit or number for example, 1 2 Apple is not a variable, valid variable name other special characters like say, Apple percentage is not a valid variable name. So, these are some of the rules. So, the what are rules of naming a C variable, it can consist of any alphanumeric character, any length, but it must start with an alphabet and only some specific special characters like the underscore is allowed, others are not.

So, this is not allowed, but say Apple bob 3 4 is a valid name. So, here I can use as again a p p l e b small o, small b, there is also a valid name. So, in general what is the rule; I can have a string of alphanumeric character, starting with an alphabet and having no

special characters except for this underscore all right. So, with that, that is how we will name the variables and each variable I will again repeat, you know that a variable essentially consists of memory locations, a variable is nothing, but a memory location and we are putting the name to that particular memory location.

(Refer Slide Time: 25:34)



So, if Apple be this memory location, may be a p p l e is another memory location. Now, which variable will go to which memory location is decided by the compiler. So, might be Bob is another one, all right. So, these are separate memory locations.

So, variables and naming of variables is a fundamental step in writing a C program, because whatever we write, we have to write them through variables, now quickly once again we look at this that whenever we are writing suppose, again I am writing.


(Refer Slide Time: 26:28)

A, B, C, AVG

$$\text{AVG} = \frac{(A + B + C)}{3}$$

← - Sentence

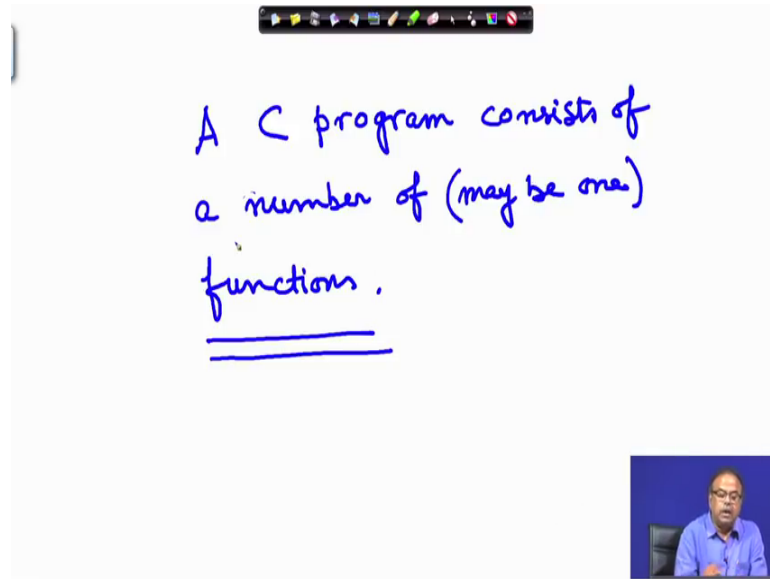
← - statement



Now, suppose there are three numbers A B C or. So, I want to find the average of that. So, I can write it in the form of say and say average, you can see that average is the valid name in the case of a variable. So, I can say A V G is A plus B plus C, whole thing divided by 3. Now, this one I can call to be A C sentence or I am saying sentence.

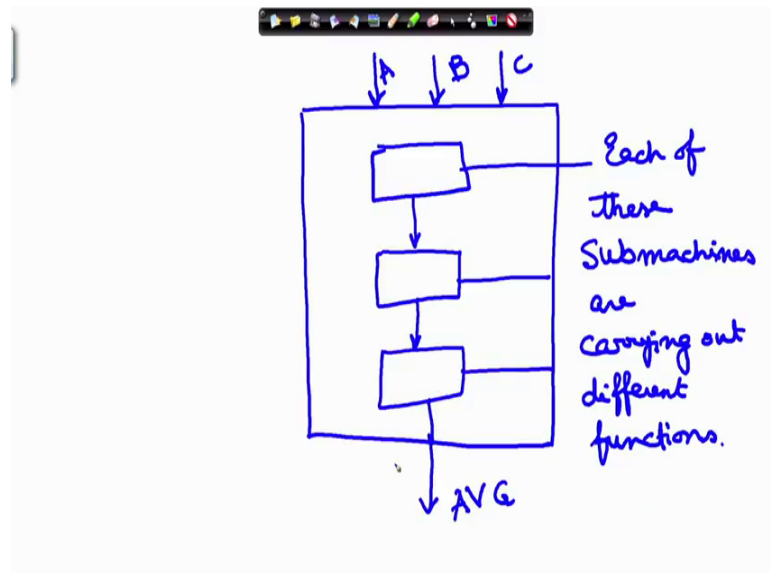
So, that you can have an analogy without English, but we will call it C statement all right and this C statement consists of several things, will come to the other things, but you can see that we are, it is consisting of 1 2 3 4 variables and one constant that is 3 all right and there are some special parenthesis and all these are also valid symbols in C and these are operators. So, we will see how was C statement is constituted, but whenever we write A C statement, it will consist of a number of variables.

(Refer Slide Time: 28:12)



Now, coming to A C program, A C program consists of a set of in, let me call it a number of may be one, but at least one number of functions. What are functions? There is a significance of this term functions, but for the time being let us simply try to visualize it in this way that I am trying to build a machine.

(Refer Slide Time: 29:17)



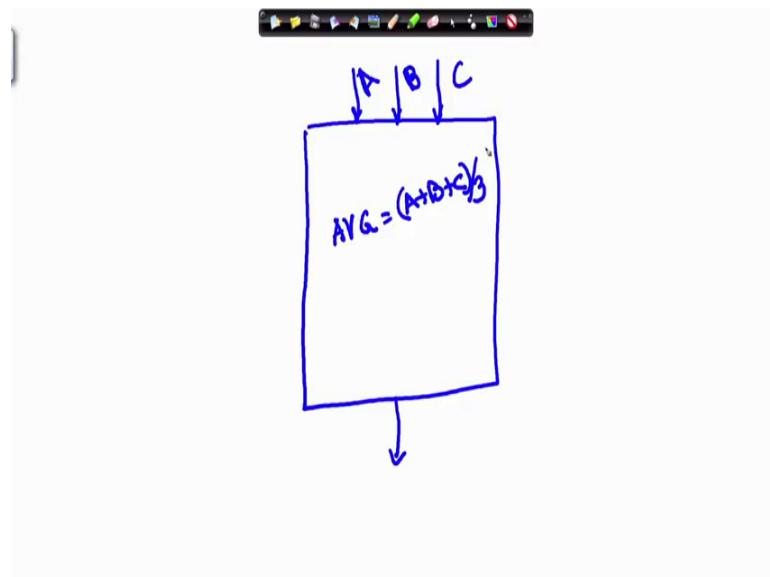
Which will do something, what will it do.

Say for example, it is a machine that will take, say 2 or 3 numbers, A B C and will produce the average. Now, this is the machine all right. It does something; takes some

input and give some output. Now, this entire machine can be built of with smaller sub machines all right and one machine, each of this sub machines can do some specific task for example, if I take a complicated, say paper rolling machine, then there are so many things to be done in order to roll out or prepare papers.

So, similarly, there can be different sub machines, which are doing different functions. So, each of them, each of these sub machines are carrying out different functions are carrying out different functions, all right. This is doing one, this is doing another and all these three together is doing something.

(Refer Slide Time: 31:05)



Now, as I said in the worst, in the, in a special case just like for a simple case of average, I may not need many sub modules, because I can very well take A B and C here, A B and C and simply in this machine I can write a program like A V G is equal to A plus B plus C divided by 3 and that will be the output right.

So, here I do not need any sub machines, because it is simple problem, but never the less, I need at least one function, that is what is this task is being done. So, any C program will require at least one function and in many cases we will see, it will require more number of functions. We will come to this in the next lecture.