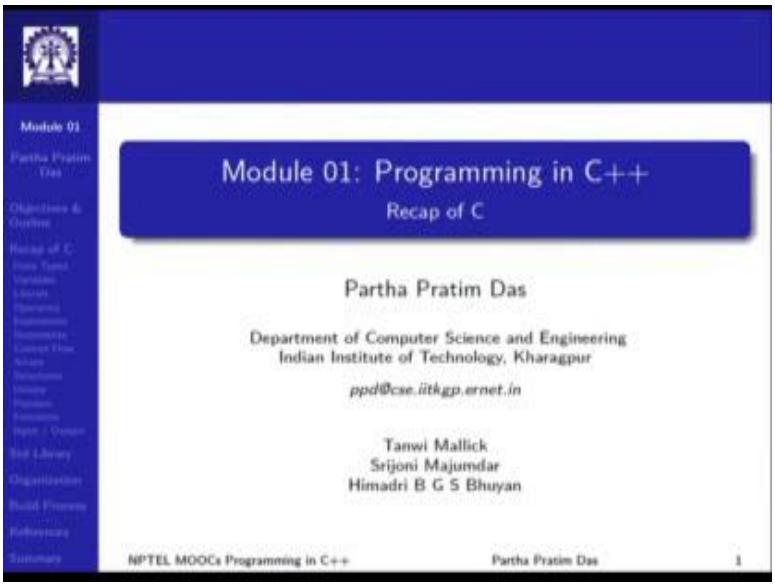
Programming in C++
Prof. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 01
Recap of C (Part I)

Welcome to programming in C++. This will be 20 hour course, where we will talk about various aspects of the C++ programming language and it will be divided naturally into about 40 modules that you will study one after the other.

The main emphasis of this course is to teach, how C++ programming language should be used in designing and implementing complex software systems.

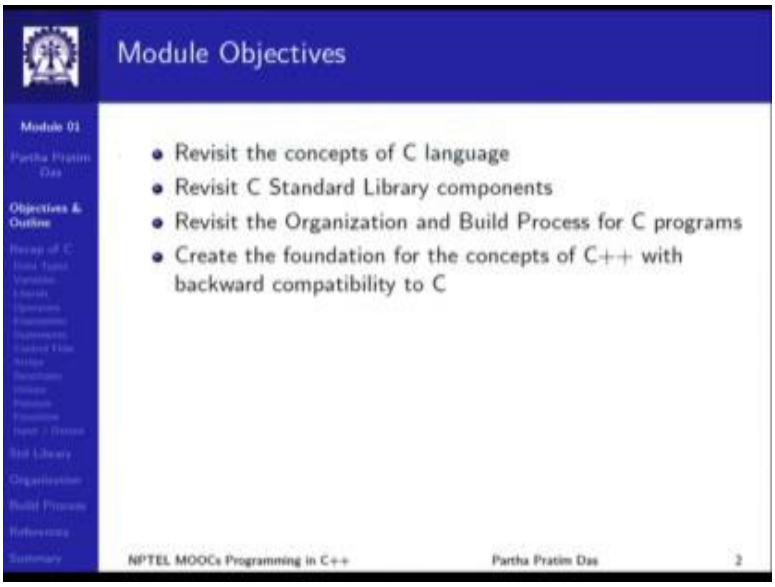
(Refer Slide Time: 00:57)



So, you all will be aware that C++ is object oriented or object based programming language and I would assume that you know C language, may not be at a very depth, but you have the overall idea about the C language. So, we will start from there, in the module 1, we will primarily talk about recapitulating various specific aspects of C programming. This is just to make sure that you can, if required you can revisit those concepts and before we get deep into the C++ language, you can be familiar with all the

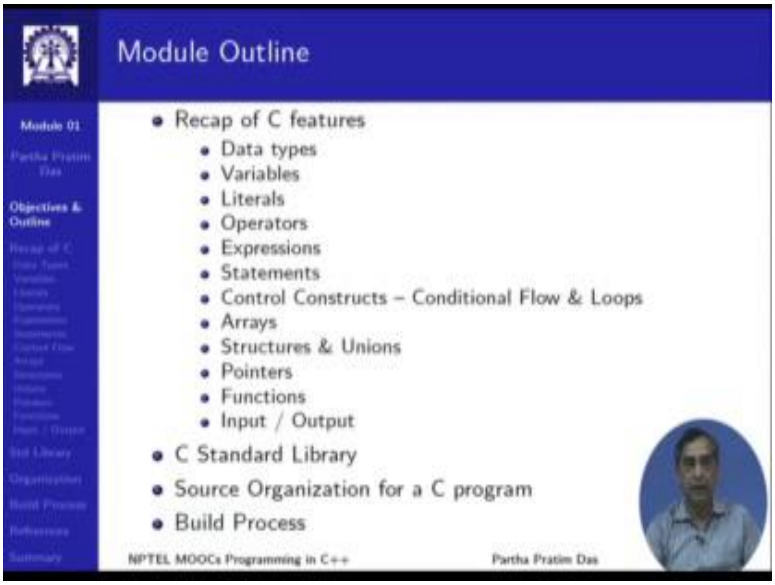
C programming requirements because C is a language, which is backward compatible to C++. So, we will first get started with recapitulation of C.

(Refer Slide Time: 01:56)



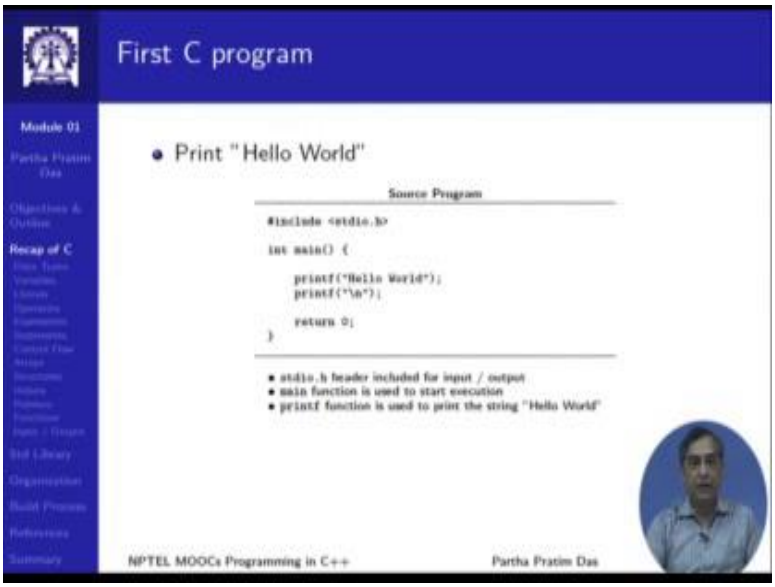
So, these are objectives to revisit the concepts, particularly we will look into C Standard Library, besides the C language and programming aspects. We will briefly discuss about the organization of C program, how C program is to be organized possibly. So, far you have only written code in terms of one single file using possibly 1 or 2 functions only. One of them must be main, as you know, we will show how to organize programs better and with this we will have a foundation to; for the C++ programming language.

(Refer Slide Time: 02:36)



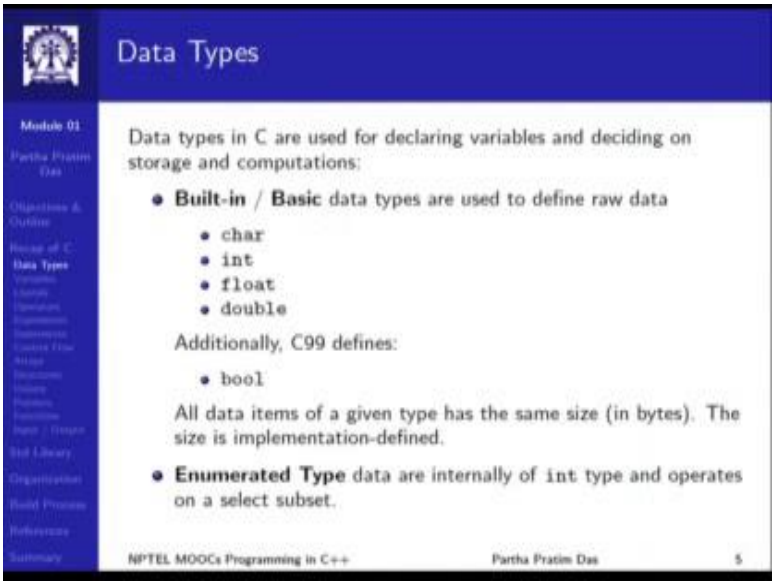
These are the different topics to be done, the outline of the module; this is for reference. As the presentation, we will proceed on the left of your screen, you will see this outline and it will be highlighted as to which particular topic we are talking about.

(Refer Slide Time: 02:54)



So, this is the first program “Hello World” which I am sure all of you have studied. This is also the starting program in Kerning and Ritchie’s famous book. We use ‘printf’ from the stdio library and print the hello world on to the terminal or which is formally set to with the stdio out file. The main function is one that you can see here is where the execution starts and then you print this string and print backslash n, which means you basically go to the next line; new line.

(Refer Slide Time: 03:37)

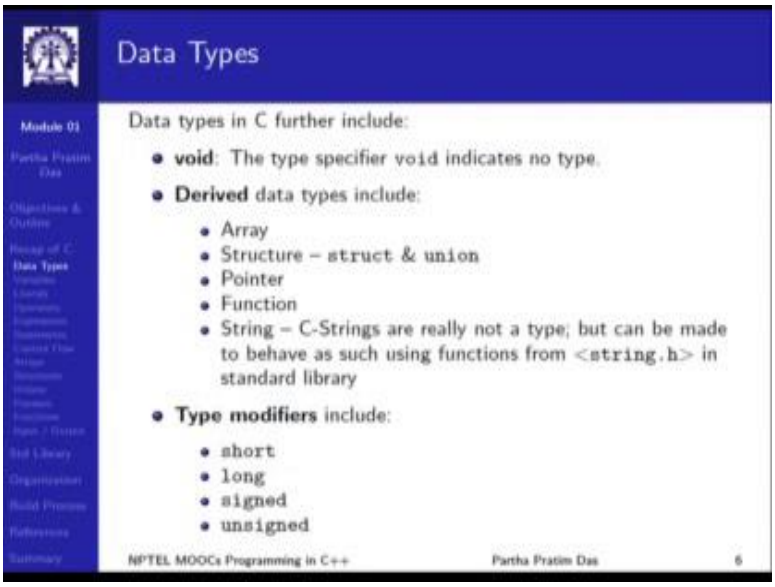


C has a number of data types. Those are known as char, which is character; int, float and double; float and double for the floating point numbers and integers are for the so called whole numbers.

Now, here I should mention that the C that you commonly use is known as C89, C89 is a first standard of C that was created by ANSI, the standardization organization and subsequently in 99, another standard was released, this is called C99, so most of the compilers today follows C99 standard. We will also expect C99 to be followed. So, when we talk about C, we will try to highlight, if few things have become different in C99. So, in terms of data type as you can see, there is a new data type bool, which has got added in C99.

In C89, you could still have Boolean values, which can be true or false based on it being an integer value. So, if it is 0, it is false; otherwise it is true. But in C99, there is a separate type bool. Every data type as you know, this built-in data types has a size that is given in bytes and you can use size of operator to get that. You can define enumerated types which are basically integer values which are given some symbolic names.

(Refer Slide Time: 05:17)

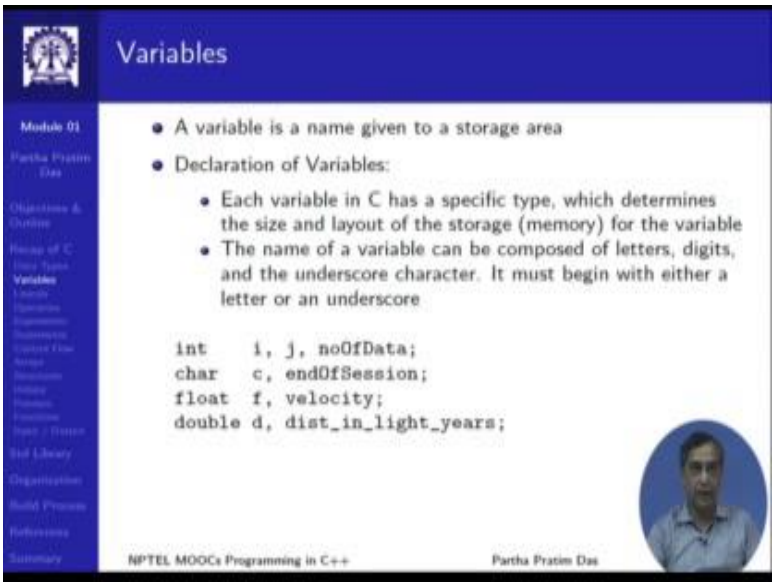


Other data types in C include void. Void is not a type, it is very interesting use and as we go into C++, we will see various different use of void. Void is where you would need to use a type, you can use the type void, but it actually says that there is no type. So, it is like, when we do arithmetic we have a 0. So, I can add 0 to x and it does not change x. So, as we say every system needs a 0. So, void is a 0 of the type system, as we will see more in C++.

Then based on this built in types, there are various derived types that supported the array, structure and union, pointer; we can have functions and it is the commonly called, there is a string type in C called C strings these days. Very strictly speaking string is not a type in C, you will understand that more when we go into C++. C strings are actually a collection of functions in string dot h header, which allow us to manipulate strings in C.

Finally, the data types can be modified for their size and whether they will be signed or unsigned and these 4 type modifiers are used in C.

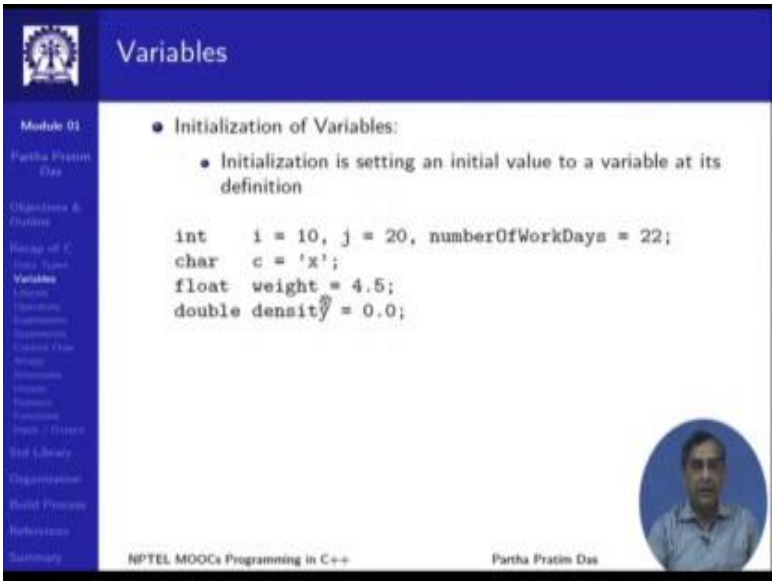
(Refer Slide Time: 06:42)



We will move on there are as you know the variables in C. Variables have certain, their names can be defined in certain ways starting with an alpha or an underscore and then extended with alpha numeric.

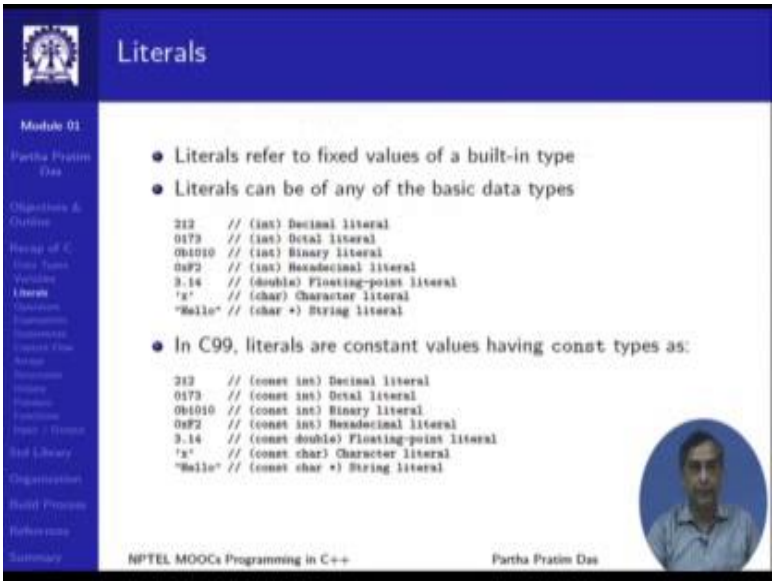
Here are some examples of different variable names, while it is often convenient to name variables with single letters or 1-2 letters. It is advised that you use variable names which make some meaning. So, we are saying corrected end of session, you could have just called it c or d or a, but it is better to give it in name from which it can be understood as to what the variable means.

(Refer Slide Time: 07:28)



When the variables are declared as they are declared here, then the variables can be initialized also. That initialization is optional. So, when we say int i initialized with 10. It means that i is an int type variable whose value at the point of definition itself will become 10. So, if you do not give initialization then it is uninitialized variable which will have an unknown value to start with. Certainly, it is very good to initialize all variables that we declare and define.

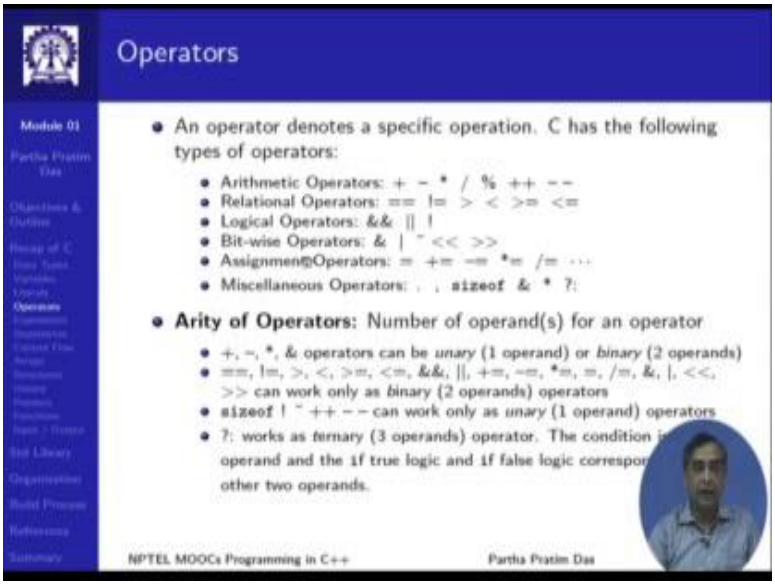
(Refer Slide Time: 08:05)



C has a number of literals which are basically fixed values of built-in types depending on how you write a particular literal, the type of that literal is decided. For example, if you just have a sequence of numerals then it becomes a decimal integer type, but if you prefix that with 0, then it is considered to be an octal type, a base eight number. If you prefix it with 0 x, then it is considered to be a hexadecimal literal and so on. Character literals are within single quotes and string literals are within double quotes.

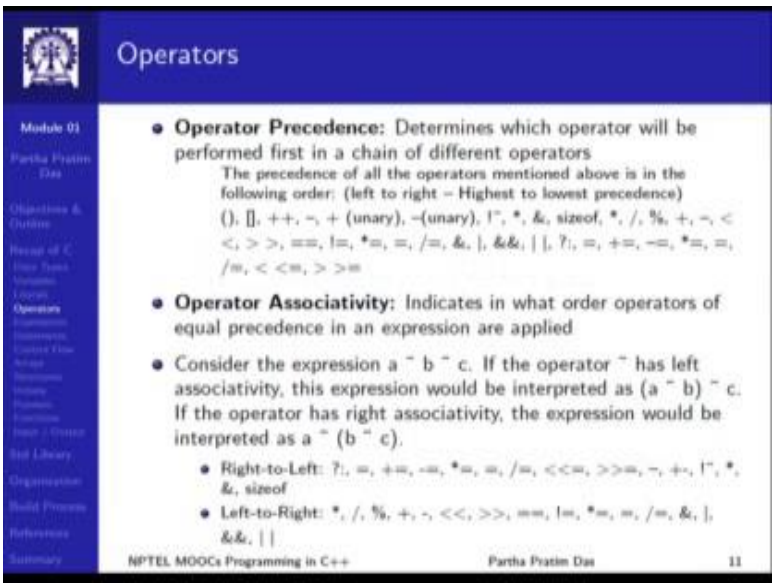
With C99, we have the introduction of what is known as const types, that are constants and we will have more discussions of that in depth when we do C++. So, in C89 the literals are basically fixed values, but in C99, they are considered to be constant type data. So, 212 in C99 it will be considered a const int.

(Refer Slide Time: 09:14)



There are several operators in C; you will be familiar with many of them. There are by common or binary operators like addition, subtraction, multiplication. There are unary operations like negation. There are even ternary operations like question mark, colon. Every operator has a fixed arity that is a number of operands that it takes, which could be 1, 2 or 3.

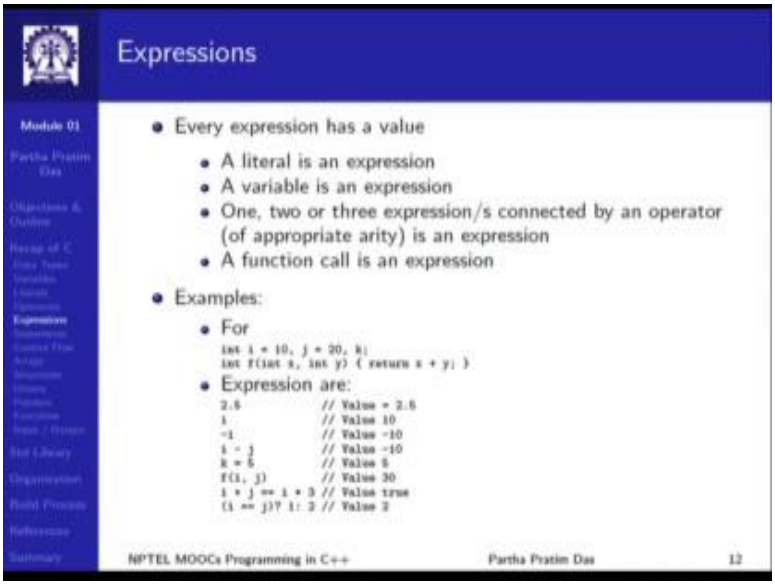
(Refer Slide Time: 09:46)



The operator in an expression is evaluated according to their order of precedence. Some operators have higher precedence, some have lower precedence. So, we know that if in the same expression there is multiplication as well as addition; multiplication has to be done earlier wherever it occurs in the expression.

Similarly, if there are more than one of the same operator in an expression then the order of their evaluation will depend on the associativity and some operators are left to right, some operators are right to left. So, here I have shown the different examples. This is just for your reference, you will certainly, if you know this. If you do not, please look up the text to understand this well.

(Refer Slide Time: 10:30)

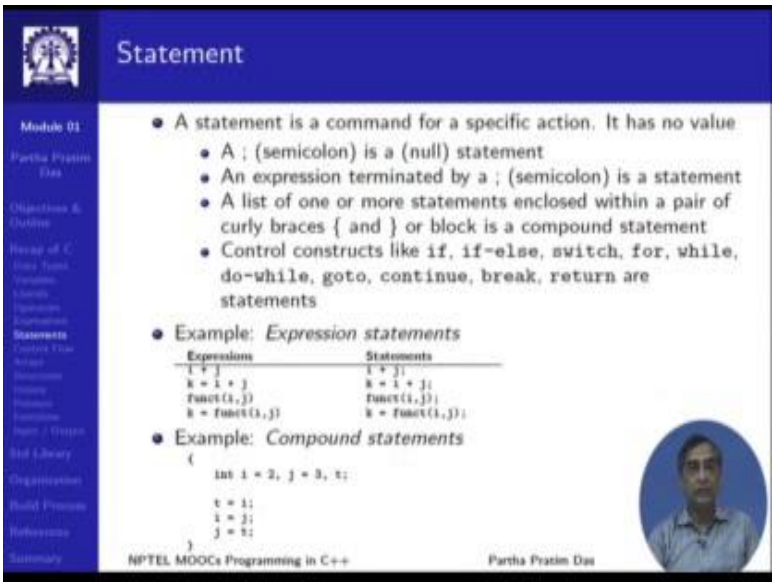


Now, the next concept in C is an expression. That I have variables and I have operators. I have literals with those I can build expressions. So, expressions are defined in kind of a recursive form will say that every literal is an expression.

If I say number 5, it is an expression by itself. Every variable is an expression and if I have two expressions and connect them by a binary operator then that becomes a new expression. Similarly, I can have expressions with unary operator, ternary operator and so on. Any function call that is done is an expression. So, the basic point is that expression

must have value; anything that has a value in C is called an expression. So, there are different examples you can see here for the variables given here and different expressions are given below.

(Refer Slide Time: 11:34)



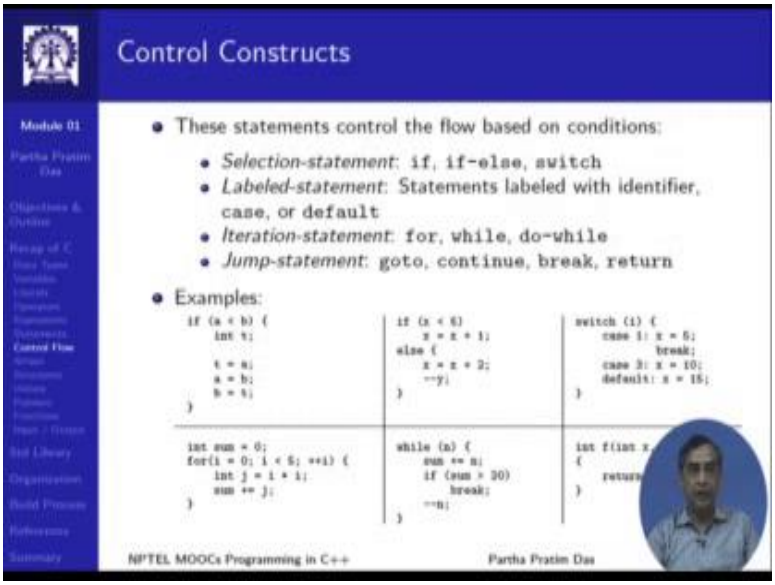
Now, expressions cannot exist in C by themselves. So, expressions will have to exist as statement. A statement is a smallest unit of command that you can specify in a C program. So, the simplest or the smallest statement that you can have which is called a null statement is a semicolon itself. Otherwise, if you have an expression, you can terminate that with a semicolon and once you terminate it with a semicolon then it becomes a statement.

So, if you look at the example below, in the expression statement i plus j is an expression because i and j are variables and plus is an operator connecting them, but the moment you write i plus j semicolon, it becomes a statement. It can occur independently anywhere, similar examples are shown for assignment for function call and so on.

Besides the expression statement, C has a number of control statements or control constructs, which basically allow the control flow in the program to be managed. So, there are selection statements and loop statements and so on. We will see little bit more

of them in the next slide and if there are number of statements one after the other which need to be grouped for use, and then we put a pair of curly braces around them. We say it becomes a block and such a statement is called a compound statement. So, that whole block of statements is a compound one you can see an example at the bottom.

(Refer Slide Time: 13:19)



Now, coming to control constructs which are the key area of a C program which basically tell you, how the execution of the program can happen. We have different ways to control, what will be executed after one statement has been executed. By default we say, the C program has a fall through control, which means that once a statement has been executed then the immediately next statement in the program code will be the next statement to be executed, but we can change that by the control flow.

So, the first kind of control flow is a selection statement; ‘if’ or ‘if else’. So, in the example as you can see that we are saying, if a is less than b, then if that is true then you do the compound statement that follows it. You can easily understand that what the compound statement is saying that you interchange the value of a and b by using a third variable.

If you look at the next example of if, it is showing ‘if else’ kind of statement, where if x is less than 5 that if the condition is true, it does one statement, else if the condition is false then it does another statement. You can see on that the false part has a compound statement, whereas the true part as a single statement and selection can be done for a multi way, in their multi way form that you can use the value of a variable.

In this case we have used the variable i and you can switch on that depending on what value the variable has taken, you take any one of the cases that are listed. So, if i is one then case one will be selected by which x will become 6 and we have a default case which is executed, if the value of i does not fall among the different cases that exist. Statements like case as we have shown in switch are also called labeled statement because there is a label to them.

Then we have iteration statements where you can repeat or loop statements very commonly these are called loop statements, where you can have a ‘for’ loop which has three parts. An initial part I assigned 0, which is initially done. A second condition part which is checked every time the loop is executed and you continue in the loop provided that condition remains true and there is a body which is basically what follows the ‘for’ statement, which is the sequence of instructions or statements to be executed as a part of the loop and there is a ‘end of loop’ statement like ++ i.

Similarly, we have while loop we can have do while iteration and the final type of control statements are ‘go to’, ‘continue’, ‘break’ and ‘return’. As you know, C advises that you should not use ‘go to’. So, we are not showing example of ‘go to’, if you design a C program well then you will not have any reason to use the ‘go to’ at all. So, try to only use ‘continue’ and ‘break’ along with loop and different switch statements to achieve your control flow, but you will need ‘return’ to return from a function. So, these are the four types of different types of control constructs that exist.

To sum up what we have seen in this module so far, we have seen what are the basic components of a C program, which is how do you do a IO, how you; using the data type, how you define variables? How you initialize them? How to form them into expression

using operators? How to convert the expressions into statements and different control flow statements to control the flow of the program?

With this, we will end this part and next we will talk about the derived types and how to use the derived types in C.