**Programming in C_2.1_Operators and Expressions Part 1**

Hello and welcome to the Session. Why do we write a program? Any program is written to give a logical solution for a given problem. Now to do this, program takes a certain input data, processes or manipulates it by performing certain kind of operations and gives the output. To perform various operations, we require operators. In this session we would see the various operators that are provided by C language and how to implement them on various expressions. Let us say I need to perform an addition Operation on two variables. Let me say the variables are A and B on which I have to perform addition.

So, I would, make use of the additive operator or addition operator. So, I would say A plus B would give me the addition of A and B. Now, in this case. Plus is the operator, which tells the compiler to perform addition operation. Now, having said this, an operator is nothing but, it's a symbol which tells the compiler to perform certain kind of manipulations. Here the manipulation that I am doing is the addition on two different variables A and B. Now, what is an expression? An expression is nothing but a sequence of operant's and operators.

Here the operants are A and B, as you can see in the example and operator is the addition operator that I'm using, which is a reduced to a single value. That means to say that the result of A plus B or the addition of a plus B is stored in a single variable C. C language provides rich built in functions. It's not always, that we perform mathematical functions or arithmetic functions by making use of arithmetic operators in our program, but also we use, different operators. For example, if we have to compare two values, then we make use of relational operators. If we have to check if the value is true or false, we can make use of logical operators.

We can assign a value to a variable by using assignment operators. We can increment or decrement a value of variable by making use of increment and decrement operators. We can check if the give the given condition is true, to perform certain kind of functionality. If the given condition is false, to perform certain kind of functionality by making use of conditional operators. We can also manipulate at the bit level by using bit wise operators and also see language provides special operators like comma operators.

So, these are the operators that are provided by C language to make use of it in our program for data manipulations. C Language provides various arithmetic operators. Before using these arithmetic operators, we need to know what are the arithmetic operations that we have. The various arithmetic operations that we have our addition, subtraction, multiplication, division and also modulates operation. Now here, plus is the operator that is used for performing addition. Let us assume we have two variables X and Y, so X plus why would give the some of the expression. We can also have more than two variables. For example, we can have X, Y, Z, and the expression would be X plus Y plus Z. Next, we have a minus operator or minus symbol to perform subtraction. Then we have asterisk symbol that is, that looks like star.

We have asterisk symbol to perform multiplication. We do not use the normal multiplication symbol which looks like X or we can say into symbol. Then we have, we use forward slash for performing division. Then we have modulus operator to get remainder of two values. All these operators can be used with more than two variables. There are certain things which have to be noted before we use division operator or modulus operator.

When we are dividing two integer values and its remainder leads to a fractional value, then division operator truncates all the values after the decimal point. And also a modulus operator cannot be applied on float or double values. It has to be used on or it can be applied on integer values. Now

coming to the precedence of arithmetic operators, meaning which operator has to be given more importance and which operators have to be given less importance. By importance, I mean, which operation has to be performed first and which operation has to be performed next.

Then, multiplication, division operator and modulus operator have higher presidents than addition, operator and subtraction operator. If we consider an expression, A multiplied by B divided by C plus T then, asterisk operator, that is multiplication operator and division operator has the same presidents, but multiplication is performed first and then division will be performed because the associativity is from left to right. After performing multiplication and then division, then addition would be you know applied. Or addition will be performed onto the whole result of A star. B divided by C.

Let us consider a small program to demonstrate how to use division operator and modulus operator. The first line is a declaration statement where we are declaring two variables, months and days to be of type integer, so months and days will have or will store integer values into it. The second printf statements which displace the statement enter days onto the output console where user can enter a value. The value that is entered by the user will be read through the scan of statement where the value that user enters will be stored in the variable days. Then the next. In the next statement, we are dividing the value of the variable days by 30 by using a division operator.

Now this expression will give the quotient and that quotient will be assigned to the variable months. In the next statement. We are using a modulus operator. These days are, or days, value will be divided by 30 and remainder value will be assigned to days. The last statement simply displaces the value of months and days. Suppose we need to find a square root of a value or we need to find a log value. Then we need to make use of certain mathematical functions. So, usually the frequently used mathematical functions are to find out, a sign value or a cause value or to find out a square root value.

So, these mathematical functions are already in built in C programming and what we have to do is to include a header file called math dot hedge at the beginning of the program such that we can make use of all these inbuilt mathematical functions which are already there in the library. If we consider an expression say minus B, plus or minus, a root of B Square minus four AC divided by 2A. Now, this is how actually we write an expression. This is the way to write an expression manually, and this is the way we usually right in you know, when we're using any mathematical function or mathematical expression, we use division operator, we use a root symbol. We just say 2A but if we right the expression in the same way in program, the compiler will not understand because it uses certain mathematical functions.

So, to make sure the compiler understands the expression, we have to write it in certain way. For example, if you say 2A, it is understood by you, that 2 is multiplied by A but here to make sure the compiler understands you have to say 2 is multiplied by A by using multiplication operator. So, the whole expression can be expressed programmatically as minus B plus we use square root function to get this a square route value in to square root of B in to B. Again multiplication operator used. Minus four AC will be expressed by using minus symbol, then asterisk symbol.

So, this is the way that we have to right the expressions in the program, not the way that you usually right or use in mathematics. Let us see the various relational operators that are provided by C language. Usually, we use relational operators to compare two different values, if the value is greater than the other value or if the value if two values are equal or if one value is lesser than the other value. So, to compare to variables values, we use these relational operators. For example, if

we have to compare if X is less than Y then less than operator symbol is used. If we have to compare, if two values are less than or equal to. Meaning if X is less than or equal to Y, then we use less than equal to operator. Then if we have to compare if values are greater than each other. If X is greater than white than greater than symbol is used. If we have to compare if X is greater than or equal to Y, then greater than equal to symbol or operator is used. If we have to compare if both values are equal, that is, if X is equal to Y then, we use two equal to operators.

Now remember, if we're using one equal to operator, then it is an assignment operator. If we say X is equal to 10, the value 10 will be assigned to X But if we say X double equal to Y then, the value of X is compared with the value of y and a true value is returned if both the values are equal. In the same way, if we have to check if one value is not equal to the other value, that is, X is not equal to Y. Then we use exclamation symbol along with equal to symbol. So, these are the relational operators that we have in C language. Any relational expression will result in either a true value or a false value. That means a value one for true value and zero for a false value.

For example, in the expression, 5 are compared with 6. We are comparing if 5 is less than 6. It's a true value, so it results in one. The next expression, minus 34 plus 8 is greater than 23 minus 5 is false, so it yields 0 as its result. Then, if we see the third expression a greater than B greater than C, where A is 3 and B is 2 and C is equal to 1. Now, three is A's value and 2 is B's value, so we check if 3 is greater than 2. 3 is greater than 2 is a true value, meaning it yields in the value 1 and value of C is 1. So, 1 is greater than 1 will yield 0 because it is false. 1 is not greater than 1, but one is equal to 1. And when you are, when you have multiple relational operators in the relational expression, then the associativity of it is from left to right. Talking about associativity.

We have to know the precedence of these relational operators, meaning which operator has higher presidents than which operators and which operators have same presidents. The operators which have higher presidents and have same presidents, are greater than, greater than equal to, less than less than less than or equal to. All these operators have same presidents and are in higher presidents than equality operators. So, if we have equal to operator and not equal to operator, these operators have lower presidents than, greater than greater than or equal to, less than or less than or equal to operators. For example, if we consider an expression like 1 is less than limit minus 1. LIM is the variable that I have taken for limit.

Then, if we check the associativity, I can say, now, here it's a, Minus is the arithmetic operator that I have used and less than is the relational operator that I have used. Always we have to remember that the relational operators will have lower presidents than arithmetic operators, so the expression will be considered as limit minus 1, has the higher presidents, than I less than limit minus 1. So, the expression will be taken as I less than limit minus 1 in the bracket. Such that limit minus 1 will be resolved first and then I less than the result of limit minus 1, will be resolved next.

Then, if we consider an expression 3 greater than or equal to 2, equal to minus 4 less than 0, then we have all relational operators here. Now, when we have all relational operators here in the relational expression then the associativity is considered from left to right. That means whichever symbol comes first and is in higher presidents will be resolved. If we compare, if we see this expression. We have three operators here. One is greater than or equal to operator. Then we have equal to operator. Then we have less than operator. Now, among these greater than or equal to operator and less than operator has the same presidents, whereas equal to operator has lower presidents. Now as the associativity is from left to right, though greater than or equal to symbol and less than symbol has the same presidents, greater than or equal to symbol or greater than equal to operation is performed first and then the less than operation is performed next and then the values

air compared with the equal to operator. So, we check if 3 are greater than or equal to 2. 3 are greater than or equal to 2 is true. So, this expression leads in true value.

Then we move to the less than operator. So, we compare minus four is less than zero minus four is less than zero. Also yields to value true. So, this three greater than or equal to two will yield one minus four is less than zero will also yield one. Now we compare one with one by using equal to symbol. So, one is one which hold together is a true value. Logical operators are usually used are to yield a true value or of false value of two different expressions or two variables.

The three logical operators that C language uses are logical and logical R and logical not. Now, a logical operator is composed of double symbols. We should not be using a single and symbol or a single or symbol. It always comes with two and symbols, which is denoted by to and plus and symbols. And two our symbols, which is denoted by to pipeline symbols. And we also have, a unary operator which performs logical, not operation, and which is denoted by a simple exclamation mark. There is associativity in the expression is from left to right, so depending on which logical operator comes first, we evaluate that particular expression and also the evaluation stops once we get to know the result, whether it is a true value or a false value. This table gives us the logical operators and their operation.

Now, when logical and operator is used with two variables X and y, then. If both are non-zero values, or if X and Y are non-zero values, then the result will be true. Otherwise, even if one of the variables is zero, then it will, the value will be false. Then we have X logical or Y, which will return false when both the values are non-zero, otherwise it will return a true value now. Not logical, not X will return true value if it is non-zero. Otherwise, it will return false. The precedence, if we compare the precedence of logical and, and logical or operator then logical and operator has higher presidents than the logical or operator. Now, next, the expression here is, you know is checking to decide whether both the, characters A and Z are in upper case or not. If CH is greater than or equal to Capital A and CH, is less than or equal to capital Z. Now, if both are non-zero, if both the expressions are non-zero, then the result will be true. Otherwise, it would be false. Thank you and we have reached the end of decision

**Programming in C_2.2_Operators and Expressions Part 2**

Hello and welcome to Session. As we know that C provides various different operators, like arithmetic operators, relational operators and logical operators. It also provides, other operators for manipulation of data such as assignment operators, increment and decrement operators, size of operators. In this session, we will also learn about the presidency of operators. To assign a certain value to a variable or to expression an assignment operator is used. Assignment operator is usually denoted by an equal to symbol. As we know this, C also provides short hand assignment operators, which looks like V operator that is; variable operator is equal to expression. Now V is a variable here, and operator O. P is an operator, which is, along with, which is combined along with equal to symbol. Then we haven't expression. This is a shorthand assignment. For example, X plus equal to five is expressed as X is equal to X plus five. X is manipulated by adding five to it, and then the result is again stored back in the variable X itself. So, such kind of operations can be performed by using shorthand assignment operators.

As mentioned in the example the assignment statement V that is variable operator equal to expression is same as variable equal to variable operator expression. So, this assignment statement can be expressed or is equal to V is equal to V operator expression that is variable equal to variable operator expression, which can be written by using shorthand assignment as we operator equal to

expression. For example, the expression or the assignment statement X plus equal to Y plus one, where X is variable, Plus is the operator plus equal to is the shorthand assignment operator, Y Plus one is the expression, is same as X is equal to X plus Y plus one. This table provides various shorthand assignment operators. For example, X equal to X divided by Y can be expressed as X division operator equal to Y.

So, in the same way, X is equal to X plus Y can be expressed as X plus equal to Y. So, these are certain shorthand operators, shorthand assignment operators that are used in various expressions. While programming sometimes we need to increments a value of a variable just buy one. In such case, we can use increments operators and decrement operators, which are urinary operators that mean it's a single operator.

Increment operator increases the value of a variable by one and decrement operator decreases the value of the variable by one. So instead of writing X equal to X plus one, we can just write or express it as X plus. Now this increments operator and decrement operator can also be used either before the variable or after the variable. If we use it before the variable, then it is called as a prefix. If we use the operator after the variable, then it is called US Post fix. Consider an example by assigning a value, integer value five to a variable A and also observe what happens if we use this increment and decrement operators on this variable A. Now the increments and decrement operators can be prefix and post fix, which, actually increments and documents the value of variable by one. Now, when I say plus A, then the value of A becomes six. Now the value of a six. Then when I say EA plus, then the value of a become seven. Now because the value of a seven when I discriminate it by saying minus A, the value of a become six. Again when I say A minus minus, the value of becomes five.

Now, when we know the increment operator increases the variables value by one and decrement operator decreases the value by zero, then why uses them as prefix and post fix. Using increment and decrement operators, either as prefix or as post fix has a definite meaning. Now, when we use this increment operator as a prefix with the variable, then the value of the variable is incremented first, and then the value is returned. And when we use increment operator as post fix, then the value of the variable is a returned first and then the value will be incremented. This holds good for even the decrement operator. It exactly works as the increment operator, but here in the decrement operator, the value will be decreased by one.

So, when used as a prefix operator, then the value will be discriminated first and then the value will be returned when used as post fix operator. Then the value will be returned first and then will be decremented. Let us see an example which demonstrates the increment and decrement operators here. In this example, we have used only increment operator. We can relate it to decrement operators too. In the first line, we have declared two variables. Which are one which is variable one and variable two. Both are assigned to same values, which is five. Now here in the printive statement, it's a display statement we have used, an increment operator, which is post fix. That is increment operator is used after the variable.

Now when increment operator is used as a post fix operator, then the value will be returned first and then the value of the variable is incremented. Here in the printive statement, we're using a format specifier to display the value of variable. Now this value of the variable is returned first to the format, specifier percentage D and then the value will be decremented. So, in this display statement, when we say percent D variable one plus plus the value of variable one being 5, 5 is returned to the format specifier percentage D first. That is the reason why it is displayed first as why and then the value will be incremented. Now again, if you print the value of variable one, then it would be six. Then in the next printive statement, we have used same format, but we have used variable two. The

value of variable two is also five. But here we have used prefix operator. Now, when we use prefix increment operator, then the value will be incremented first and then the value is returned. When the value is incremented, variable to will be incremented to six as its previous values five and then the value is returned to the format specifier percent D and thus we have six as the output.

Now the question which would arise can be, can we use this instrument and document operators along with expressions too? The increment and document operators cannot be used along with expressions, but these are applied only to variables. So, using it with the expression I plus J, plus plus is not a valid expression. We should not be using, the increment and decrement operators along with their expression. Now, let us consider on example M is equal to M plus plus minus J plus 10. We can use increment and decrement operators along with the expressions that are we can include these increment and decrement operators, in or within the expressions. So we can have, variable N, which can be post incremented.

Then we can have minus J plus 10. Now what happens to the presidents of the operators when they are used in the expressions? Now, the pre the increment and decrement operators have the same presidents as of the normal unary plus and unary minus operators. But the associativity of the expressions will be from right to left. If you consider an expression M is equal to minus M plus. Here we have used to operators. One is minus, which is a unary operator, and other is the increment operator, which is a post increment operator. So, this is equal to, M is equal to minus M plus, because now, as a, plus and minus has the same presidents that is post increment and minus operator has the same presidents.

But the associativity of it would be from right to left. N plus will be evaluated first and then will be associated with urinary minus operator. We also have a size of operator, which is a compiled time rob operator, which, when used along with an operand would return the number of bites that the operand has occupied in the memory. Now the operand can be a variable or it can be a data type or it can be a constant. For example, you want to know how many bites that does an integer data type in your machine has been allocated with. You can use this size of operator to check the size of that particular data type. Here in the example, we have used size of longant, so this would return the number of bites that are occupied by longnat type qualifier in your machine and will return that to N. In the above example, when you have M is equal to size of sum. Sum is a variable. It can be of any data type. If it is of integer data type, then size of sum will be returned with two bites, as most of the machines would allocate two bites to integer data type. If it is care that is the variable sum is declared to be a care data type.

Then it will return one bite of data and the value will be returned to the variable. And this program simply demonstrates the usage of size of operator. Here in the first line of the program, we have declared a variable of type integer, the variables name is I and is assigned with the value 10. Now, if I have to know the number of bites the variable I have been allocated with, then we can use the size of operator. This size of operator will calculate the memory that has been occupied by variable I and returns the value to the format specifier percentage D here. So, size of operator here is used to calculate the number of bites that are allocated in the memory by using on operand or a constant or a data type. The output of this program will be two bites because most of the machines again will assign two bites or allocate two bites to an integer data type. As we use different operators in the expression, these operators might be at the same level or might be at different levels. That means higher and lower levels.

Now the presidency rules decide the order in which these different operators air applied associativity rules the order in which the same level of operators are applied. The table here

expresses the associativity, of these operators which are at the same level. For example, if you are If we consider the first row operators which are opening brace closing brace, we have an arrow operator and we have a dot operator. All these operators are, having same level, or are at the same level. And if, there are multiple operators which are at the same level are occurred in the expression, then this it's associativity would be from left to right. Okay. And for example, if we have an increments operator are, a unary plus operator, then in an expression, then it's associativity is from right to left.

So, these are all the associative it is that we can have along with the operators which are at the same level. We have reached the end of session.

Thank you

**Programming in C_2.3_Operators and Expressions Part 3**

Hello and welcome to Session. In this session you will be learning about how to implement conditional or a turnary operator. How to include library functions in our program, how to manipulate data at the bit level by using bit wise operators and also a special operator called as a comma operator. A turnary or conditional operator uses a question mark symbol and a colon symbol to construct conditional expression. And, there are three expressions which are used here. All the three are conditional expressions. Now, expression one and expression two is separated by a question mark. Expression two and expression three is separated by a colon, upon the true or false value of expression one.

Expression two, either expression two our expression three will be evaluated. If expression one is true, then expression two will be evaluated and expression three will, evaluation of expression three is skipped. If expression one is false, then expression three will be evaluated and the evaluation of expression two will be skipped. They include various are commonly used operations or calculations in our program which are already inbuilt. So, we happen to use all these inbuilt functions are a library functions in our program. Now, certain library functions that we use are the standard input output operations, which are like printf and scanf functions.

We use various operations on characters or strings. Or we even use certain functions which carry out mathematical functions. We have many library functions which are similar. So, what c language does is it groups of all the similar library function under or as one object program in separate library files. For example, if we have many inbuilt mathematical functions, all these mathematical functions are group together under one header file, which is called us math that math dot hedge. Now, if we want to use, input output, library functions, then we have to include a header file called as STDIO dot H. Now, how do us access this library function is that we just have to write the function name and its associated parameters.

And where does this function return a value? It returns a value where it is assigned to. It can be a constant or it can be an identifier. It could be anything. This table gives a vague idea of the library functions, which we mostly use. Not all the library functions are mentioned, but few of the library functions which we can be invoked, for example, to upper is a function that we invoke to convert a character or a letter in to an uppercase value. We can also use in the same way to lower function. So, to get a tangent value we use a tan function to get a log rhythmic value, we might use log function. We can also get ceiling and floor values by using seal and floor functions.

So, these are all the library functions that we can use in our program. This is a simple program to demonstrate how to include library functions in the program. This program is basically converting a

lowercase character in to an uppercase character. So, the first line hash includes STDIO dot hedge. Includes all the input, output functions which are available under that header file. We also have a C type dot hedge, which includes all the character functions that are available. So, coming to the main program, the first line int lower comma upper will, declare two integer variables which are lower and upper. So, here in the next line lower is assigned with getchare function.

Now getchare function is a function which reads the character from the standard input that is here a keyboard so you can press any key value. Let it be A, B, C, D, and E, F or anything which will be stored in lower. Now when any character is stored in an integer variable, then its asking value is stored and in the next line, upper variable is assigned to; value where this lower variables value is converted to upper by using to upper function. So, supposing I am sending or I'm pressing the letter small ‘a’. Then that ‘a’ will be assigned to lower variable and lower variable here in the third line is taken as a parameter along with the library function to upper, so to upper will have the value of the lower value.

Here in this case, it is small ‘a’. That small ‘a’ will be converted to capital A with the help of two upper function and that capital A will be assigned to upper variable, which is, user defined variable here. And to display that particular character onto the console. We use put care function. Here, put care function, displays the value onto the standard output, which is our output console. So the value of upper is A, capital A that would be displayed on to the output console. This is again one more function to illustrate all the library functions. Unlike the other program where we have used C type dot hedge for character functions. Here we are, including math the dot hedge for using mathematical functions.

So, any header file can be included. Any library functions can be included by, it's associate, by including its associated header file. So, if we're using mathematical functions, then it is, mandatory that we have to include that particular header file. And if we're using standard input output functions, then we have to include standard input output header file. So, here in the program, the first line where integer variables, there are three integer variables which are declared and assigned together. So, I is an integer variable assigned to minus 10.

E is an integer variable again assigned with value to, D is assigned with value 10. And you also have a float value, which is RAD, which is assigned with value 1.57. You have to double values D1 and D2. Now, here in the display statements were using various mathematical functions and displaying the values of you know the numbers or the variables that we're using. Here, absolute I are the mathematical function that we're using. So absolute function, takes the value I that is minus 10 and then converts it into an absolute value. Then displace or returns the value to the format specifier percent D. So, this is how we use all the mathematical functions here.

As we know, all the values are considered by the machine as binary values. Whichever integer values, you store will be converted to a binary number. And then, you know, the various operations are performed. To perform operations at the bit level, we use bit wise operators. We have various bit wise operators. We have an operator which performs bit wise and operation. We have or operator which performs bit wise or operation. We also have a negation which performs bit wise, not operation that is all the binary bits will be complemented.

We have an XOR Operator which performs xor operation. We have a left shift operator which performs left shift operation. That is, it shifts all the binary numbers to one position left. And when we're using a right shift operator, it shifts all the bits to one position right. We also have a special operator which is called as a common operator. Now why we use comma operator is to link the

related expressions together. Now, for example, if we have to declare to integer variables and we have to assign the values also at the same time. To do it in a single statement. Now we can do it as shown in the example here where we have int A is equal to 10 Comma B is equal to 20. Now A is equal to 10 is one expression and B is equal to 20 is another expression. Now both of them can be linked together by a comma operator here. And the execution here is done one after the other. First, A will be assigned with 10. Then 20 will be assigned to B. So and also we can use Comma operator in fir statement where we can use multiple declarations at the same time in a single statement. When different expressions are linked with the comma operator, then the expressions will be evaluated from left to right.

For example, as we have seen when If there int A is equal to 10 comma B is equal to 20. A is equal to 10 and B is equal to 20 are separated by comma, then A is equal to 10 will be evaluated first. That means 10 will be assigned to variable A first and then 20 will be assigned to variable B. So the expressions will be evaluated from left to right and also the value of the right most expression is the value of the combined expression. For example, in the statement, that we can see value is equal to X is equal to 10; Y is equal to five X plus Y. Now all the three are the expressions which are separated by a comma. Now firstly, X is assigned to 10 and Y is assigned to five and X plus Y will be evaluated. If you can see all these expressions are assigned to a different value variable, which is called us value. So, though the associativity is from left to right, the right most expression which is X plus five, X plus Y here it i. 10 plus five, which is 15, will be assigned to the value. So, the value of the right most expression will be combined and then will be assigned to the left side value and also parenthesis are required since you have the comma operator which has a lower presidency than the parenthesis.

Thank you, as we have reached the end of decision.