CONTENT 5

Operators In C

**“Special symbols that are used to perform actions or operations are known as operators.”**

 For example, the symbol plus (+) is used to perform addition so it is an operator.

We will discuss all sorts of operator here. Let’s start with the simpler one’s i.e. Arithmetic.

#### Arithmetic operators:

Arithmetic operators are used to perform mathematical operations such as addition, subtraction etc. Few of the simple arithmetic operators are :

|  |  |
| --- | --- |
| **Operator** | **Description** |
| + | Addition |
| − | Subtraction |
| \* | Multiplication |
| / | Division |
| % | Modulus |

We all know their purpose and how they are used in simple mathematics. Their purpose and functionality are the same, let’s see their implementation in C.

int a = 2;

int b = 3;

printf("a + b = %d\n", a+b);

The output will be:

a + b = 5

#### Relational Operators:

Relational operators are used for the comparison between two or more numbers. Same as Java, C also has six relational operators and their return value is in Boolean i.e. either **True or False**(1 or 0).

|  |  |
| --- | --- |
| **Operator** | **Description** |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal to |
| <= | Less than or equal to |
| == | Is equal to |
| != | Is not equal to |

Let’s go to VS Code now:

int a = 2;

int b = 2;

printf("a == b = %d\n", a==b);

The output is 1 i.e. True.

If we change the value of a or b the value will be false or 0.

int a = 1;

int b = 2;

printf("a == b = %d\n", a==b);

The output is 0 i.e. False.

#### Logical Operators:

There are three logical operators i.e. AND, OR and NOT. They can be used to compare Boolean values but are mostly used to compare conditions to see whether they are satisfying or not.

**AND**: it returns true when both operators are true or 1.

**OR**: it returns true when either operator is true or 1.

**Not**: it is used to reverse the logical state of the operand.

|  |  |
| --- | --- |
| **Symbol** | **Operator** |
| && | AND operator |
| || | OR Operator |
| ! | NOT Operator |

**Example:**

int a = 1;

int b = 0;

printf("a or b = %d\n", a||b);

Here the output is:

a or b = 1

Let’s see what happens if both the values are zero

int a = 0;

int b = 0;

printf("a or b = %d\n", a||b);

the output is:

#### Bitwise Operators:

To perform bit level operations, bitwise operators are used. They convert the values we provide to them in binary format and then compare them to provide us the results.

|  |  |
| --- | --- |
| **Symbols** | **Operators** |
| & | Bitwise AND |
| | | Bitwise OR |
| ^ | Bitwise XOR |
| ~ | Bitwise complement |
| << | Shift left |
| >> | Shift right |

#### Assignment Operators:

Assignment operators are used to assign values. They are going to be used in each and every one of our program.

int a = 0;

int b = 1;

Equal to (=) is the assignment operator here, assigning 0 to a and 1 to b.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| = | Assigns values from right side operands to left side operand |
| += | It adds the right operand to the left operand and assign the result to the left operand. |
| -= | It subtracts the right operand from the left operand and assigns the result to the left operand. |
| \*= | It multiplies the right operand with the left operand and assigns the result to the left operand. |
| /= | It divides the left operand with the right operand and assigns the result to the left operand. |

#### Conclusion:

These are few of the important operators that you should know about before starting actual programming. There are also many other operators such as &, % or \*(pointer). I will let you know their details when working with them but the few defined above will be used frequently so knowledge about them is important. You do not have to remember them all as you can open the document again and use it or you also take the use of google.

As you work with more operator you will get familiar with these operator. So you need to remember them.

#### Code for explaining operators:

#include

int main()

{

/\* code \*/

int a, b;

a = 2;

b = 3;

printf("a & b = %d\n", a&b);

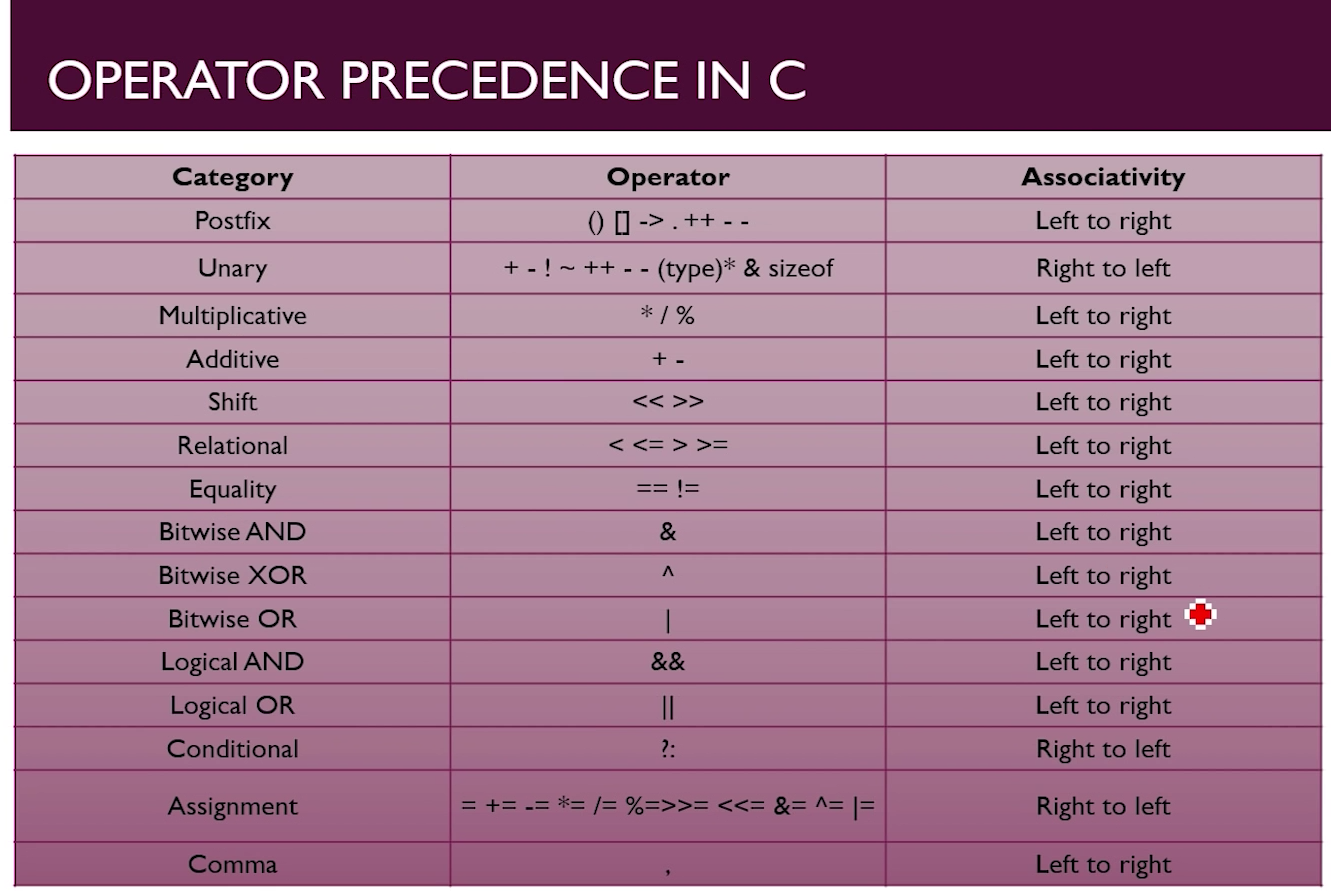
printf("a - b = %d\n", a-b);

printf("a \* b = %d\n", a\*b);

printf("a / b = %d\n", a/b);

return 0;

}



“**In Programming languages, the associativity of an operator is a property that determines how operator of the same precedence are grouped in the absence of parentheses**”

For Example;- If you want to print, 2+3\*1

Then you have to find operators in table.

As you can see operator ‘+’ is in 4th row and ‘\*’ is in third row.

Then for calculation you have to see the operator column of the table from top to bottom. As you see operator ‘\*’ comes 1st and operator ‘+’ comes after that.

The calculation for 2+3\*1 is done like; 3\*1=3

then 2+3

=5

“But If operator exits in same row (like column 4 of table) then you have to see Associativity column of row 4”.

For Example;- If you want to print, 2+3-5

Then you have to find operators in table.

As you can see operator ‘+’ and ‘-’ both are in 4th row.

Then here is the question which operator will you use first, So you to see the Associativity column.

As we go for 4th row of associativity column we can see the remark “Left to Right” which means operator at the left (+ operator will execute first) and then 2nd operator which is ‘-‘ operator (or next operator which are in random sequence in operator column).

The calculation for 2+3-5 is done like; 2+3=5

then 5-5

=0