Operators In C:-

Today we are going to learn about operators. The theory related to operators as well as showing you the code as examples. So we will be using VS Code to write a few lines of codes for better understanding of the topic. Let's start with the definition:

"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:

```
a or b = 0
```

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 Tutorial any time again when required.

Code as described

```
#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;
}
```