Logical Operators in C

Logical operators in C join two or more expressions or conditions to perform logical operations on a given expression. It is applicable in a variety of relational and conditional expressions. This operator uses Boolean values to check the condition logically and returns 1 if the conditions are true.

The logical operators in C return 0 if the conditions are false. There are three types of logical operators in C programming that are as follows:

- Logical AND(&&) operator
- Logical OR(||) operator
- Logical NOT(!) operator

Logical AND Operator in C

A binary operator is the logical AND. It combines two inputs into one. The result is true only if both inputs are true and false if one of the inputs is false. The truth table of the logical operators in C is as follows:

Input 1	Input2	Output
0	0	0
0	1	0
1	0	0
1	1	1

Logical OR (||) Operator

A binary operator is the logical OR. It combines two inputs into one. The result is false only if both inputs are false and true if one of the inputs is true. The truth table of the logical operators in C is as follows:

Input 1	Input2	Output
0	0	0
0	1	1
1	0	1
1	1	1

Logical NOT (!) Operator

The logical NOT operator is the only unary operator among the logical operators. This will take one input and return its complement as the output. If the input is true, the output is false, and if the input is false, the output is true.

Input	Output
0	1
1	0

Short-Circuiting Property in Logical Operators in C

The logical AND and logical OR operators perform short-circuiting operations, but the logical NOT operator not be used for this purpose. Short-circuiting is the process of skipping sections of code to improve a program's efficiency.

If you want to understand the short-circuit property in logical operators in C and improve the efficiency of your programme, keep these two points in mind.

When the first input is false, the logical AND operator does not evaluate the second.

When the first input is true, the logical OR operator does not evaluate the second input.

Logical Operators in C Examples

The program given below is an example of the logical operators in C that are as follows:

```
#include<stdio.h>
int main()

{

int a = 2, b = 2, c = 4, result;

result = (a == b) && (c < b);
printf("(a == b) && (c < b) is %d \n", result);

result = (a == b) && (c > b);
printf("(a == b) && (c > b) is %d \n", result);

result = (a == b) || (c < b);
printf("(a == b) || (c < b) is %d \n", result);

result = (a != b) || (c < b);
printf("(a != b) || (c < b) is %d \n", result);

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

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

return 0;
```

}

Output:

(a == b) && (c < b) is 0
(a == b) && (c > b) is 1
(a == b) || (c < b) is 1
(a != b) || (c < b) is 0
!(a == b) is 0
!(a != b) is 1

Explanation:

(a == b) && (c < b) returns 0 because operand (c < b) is 0(false).

(a == b) && (c > b) = 1 because both operands (a == b) and (c > b) are 1(true).

(a == b) || (c < b) equals 1 because (a == b) equals 1. (true).

(a!= b) || (c < b) returns 0 because both operands (a!= b) and (c < b) is 0(false).

Because (a == b) is 1, it evaluates to 0. (true). As a result,!(a == b) equals 0. (false).

Because operand (a!= b) is 0(false). As a result,!(a!= b) equals 1(true).

Problems on Logical Operators in C

Problem 1: Consider the following code:

```
#include<stdio.h>

int main()

{

int x = 7, y = 3, z = 3;
int d;
d = z + y == x;
printf("%d", d);

}
```

What is the output of the code?

A. 0
B. 1
C. 6
D. 7

Answer: A

Problem 2: What will be the output of the following C code?

```
#include <stdio.h>
int main()

{

int x = 10, y = 5, z = 3;
y != !x;
z = !!x;
printf("%d \t %d", y, z);

}
```

A. 5 1
B. 0 3
C. 5 3
D. 1 1

Answer: A

Problem 3: Consider the code:

```
#include <stdio.h>
int main()

{

int a = 10;
if (a == a--)
printf("True 1\t");
a = 10;
if (a == --a)
printf("True 2\t");

}
```

What will be the output of the following C code?

A. True 1
B. True 2
C. True 1 True 2
D. Compiler Dependent

Answer: D