

# Logical Operators

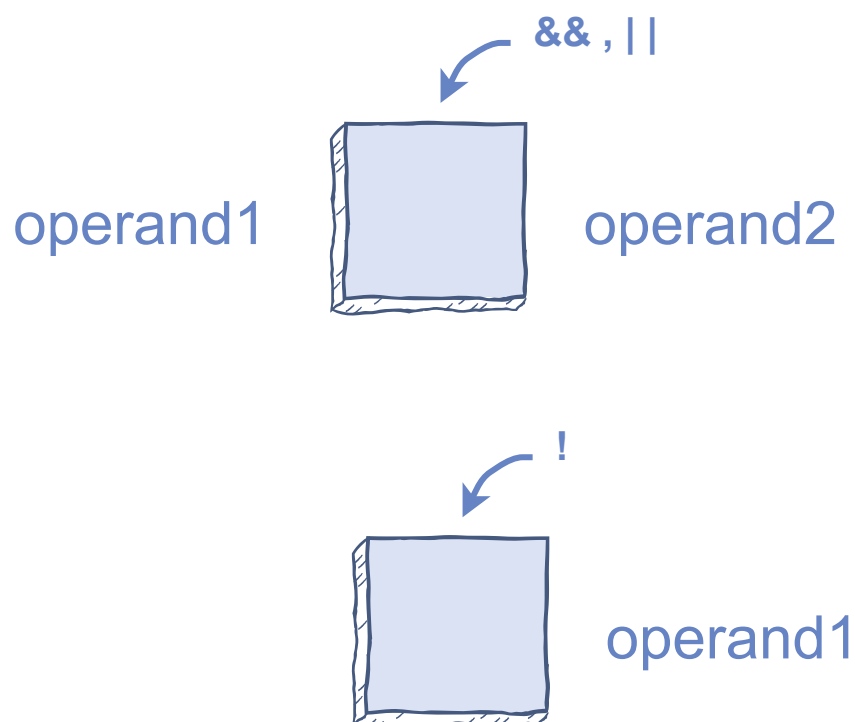
In the following lesson, you will be introduced to logical operators.

## We'll cover the following

- Introduction
  - Difference between relational and logical operators
  - Example program with bool operands

## Introduction #

Logical operators are either used to combine two or more boolean operands or to negate the result of the original boolean operand.



Here is the list of logical operators available in C++.

Operator	Operation	Use
!	NOT	Negates the value of operand1
&&	AND	Returns 1 if operand1 and operand2 both evaluates to true
	OR	Returns 1 if either operand1 or operand2 or both evaluate to true

## Difference between relational and logical operators #

Suppose there are 20 students in your class. The teacher just displayed the final result of the “**Fundamentals of Computer Science**” course with an announcement that the student with the highest marks will be given a reward. Sounds interesting! But, how would you know that are you the student with the highest marks in class?

You will compare your marks with the rest of the students in the class. However, relational operators can only compare the scores of two students and return the result. How would you compare your marks with the 19 students in the class?

Here, logical operators come to the rescue! Logical operators allow you to make as many comparisons as you want.

## Example program with `bool` operands #

Consider two operands of type `bool`: The value of `operand1` is `false` because `2` is not greater than `3`, and the value of `operand2` is `true`. The program given below demonstrates the working of logical operators.

Run the code below and see the output!

```
#include <iostream>
using namespace std;


int main() {

    bool operand1 = 2 > 3;
    bool operand2 = true;
    cout << "Values of operands are:";

    cout << "operand1 = " << operand1 << " , operand2 = " << operand2 << endl;
    cout << "operand1 && operand2 = " << (operand1 && operand2) << endl;
    cout << "operand1 || operand2 = " << (operand1 || operand2) << endl;
```

```
cout << "operand1 || operand2 = " << (operand1 || operand2) << endl;  
cout << "!operand1 = " << (!operand1) << endl;  
cout << "!operand2 = " << (!operand2) << endl;  
  
return 0;  
}
```



 Logical operators are generally used to control the flow of the program. They allow a program to decide the flow of execution based on certain conditions. We will explore this more in the upcoming chapters.

Q

What is the output of the following code?

```
int main() {  
    bool operand1 = !false;  
    bool operand2 = 8 > 9;  
    cout << "operand1 && operand2 = " << (operand1 && operand2) <<  
    endl;  
    cout << "operand1 || operand2 = " << (operand1 || operand2) <<  
    endl;  
    cout << "!operand1 = " << (!operand1) << endl;  
  
    return 0;  
}
```

[Retake Quiz](#)

---

This sums up our discussion of logical operators. Let's discuss bitwise operators in the upcoming lesson.