Bitwise Operators

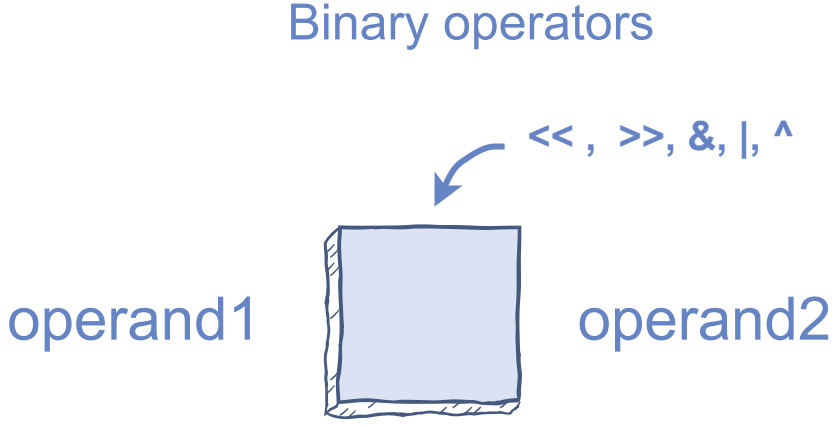
In this lesson, we discuss bitwise operators.

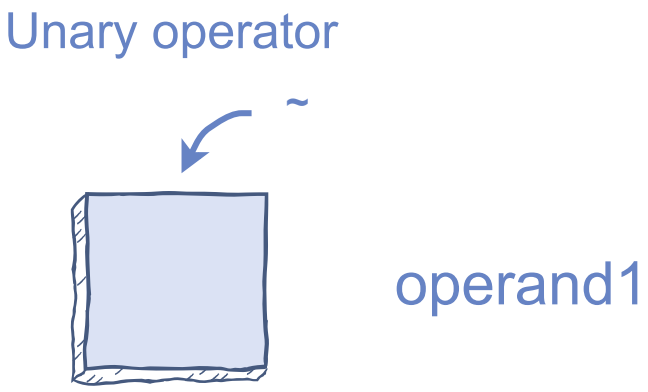
We'll cover the following

Introduction

A bitwise operator performs bit by bit processing on the operands.

Bitwise operators operate on binary numbers. They convert the operands in decimal form into binary form, perform the particular bitwise operation, and then return the result after converting the number back into the decimal form. C++ supports the following bitwise operators:





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

Operator	Operation	Use	Example
&	Bitwise AND	If the corresponding bit in both operands is 1 it will return 1, otherwise 0	1 & 1 = 1 1 & 0 = 0
|	Bitwise OR	If the corresponding bit in atleast one of the operand is 1 it will return 1, otherwise 0	1 | 1 = 1 1 | 0 = 1
>>	Right Shift	Move all the bits in operand1 to the right by the number of places specified in operand2.	2 >> 1 = 1
<<	Left Shift	Move all the bits in operand1 to the left by the number of places specified in operand2.	2 << 1 = 4
^	Bitwise XOR	If the corresponding bit in both the operands is opposite it will return 1, otherwise 0	1 | 1 = 0 1 | 0 = 1

Bitwise operators in C++

Example program with bitwise operators

Consider two operands of type `int`: The value of `operand1` is `3`, and the value of `operand2` is `2`. The program given below demonstrates the use of bitwise operators.

Run the code below and see the output!

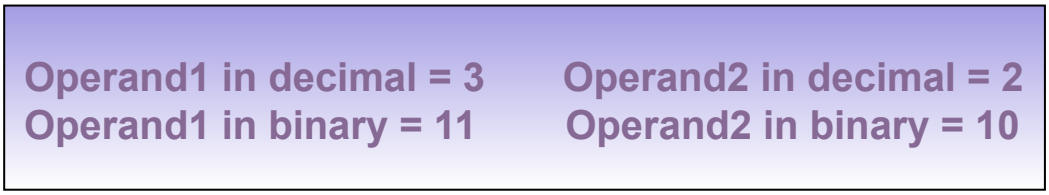
```
#include <iostream>
using namespace std;

int main() {

  int operand1 = 3;
  int operand2 = 2;
  cout << "operand1 = " << operand1 << " , operand2 = " << operand2 << endl;
  cout << "operand1 & operand2 = " << (operand1 & operand2) << endl;
  cout << "operand1 | operand2 = " << (operand1 | operand2 )<< endl;
```

```
  cout << "operand1 ^ operand2 = " << (operand1 ^ operand2) << endl;


  return 0;
}
```

&, |, and ^ operators

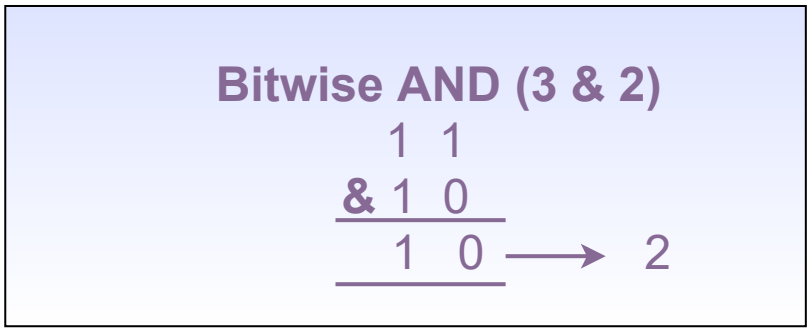
Explanation

Let's convert the given decimal numbers into binary.



Bitwise `AND`

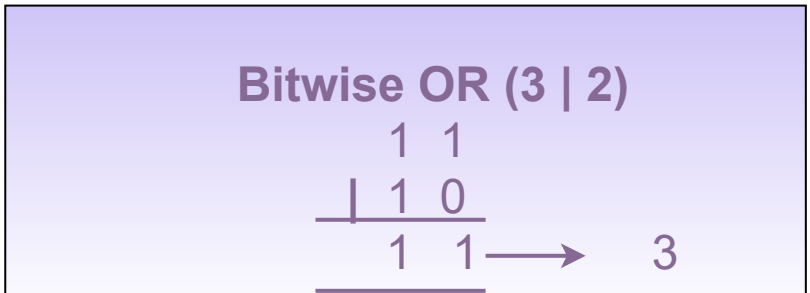
If the corresponding bit in one of the operands is `1`, it returns `1`. Otherwise, it returns `0`.



Bitwise AND

Bitwise `OR`

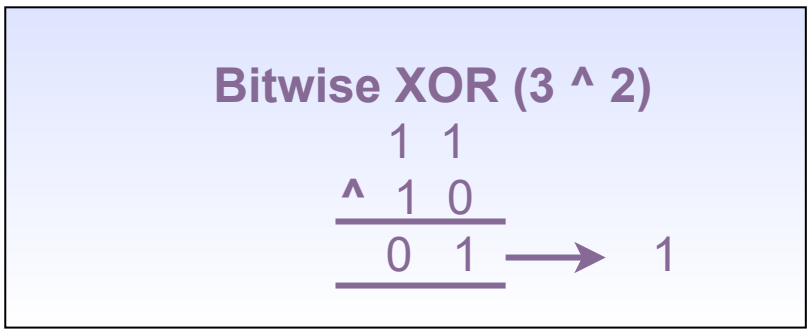
If the corresponding bit in one of the operands is `1`, it returns `1`. Otherwise, it returns `0`.



Bitwise OR

Bitwise `XOR`

If the corresponding bits in both the operand are opposite, it returns 1. Otherwise, it returns 0.



Bitwise XOR

Example program with left and right shift bitwise operators

Consider two operands of type `int`: The value of `operand1` is `2`, and the value of `operand2` is `1`. The program given below demonstrates the use of bitwise operators.

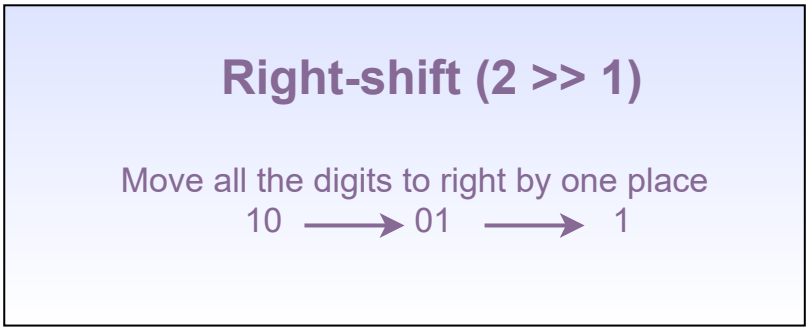
Run the code below and see the output!

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

int main() {
  // your code goes here
  int operand1 = 2;
  int operand2 = 1;

  cout << "operand1 >> operand2 = " << (operand1 >> operand2) << endl;
  cout << "operand1 << operand2 = " << (operand1 << operand2) << endl;

  return 0;
}
```

Right shift

Right-shifting an `operand1 >> operand2` is equivalent to dividing operand1 by 2^{operand2}.

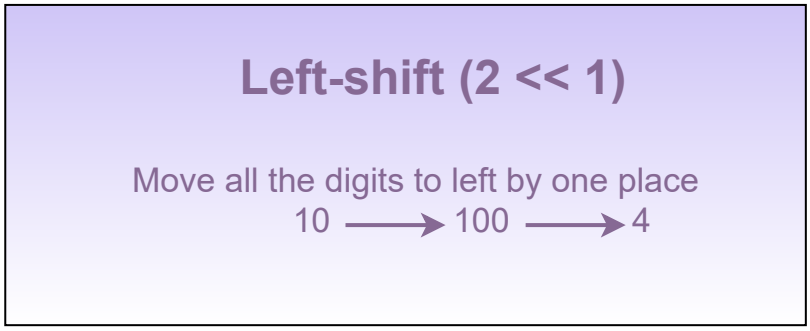
$$2 >> 1 = 2/2^1 = 1$$



Left shift

Left-shifting an `operand1 << operand2` is equivalent to multiplying operand1 by 2^{operand2}.

$$2 << 1 = 2 * 2^1 = 4$$



What is the output of the following code?

```
int main() {

  int operand1 = 6;
  int operand2 = 3;
  cout << "operand1 | operand2 = " << (operand1 | operand2 )<< en
dl;
  cout << "operand1 ^ operand2 = " << (operand1 ^ operand2) << en
dl;

  return 0;
}
```

Retake Quiz

This sums up our discussion of bitwise operators. Let's discuss the precedence and

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