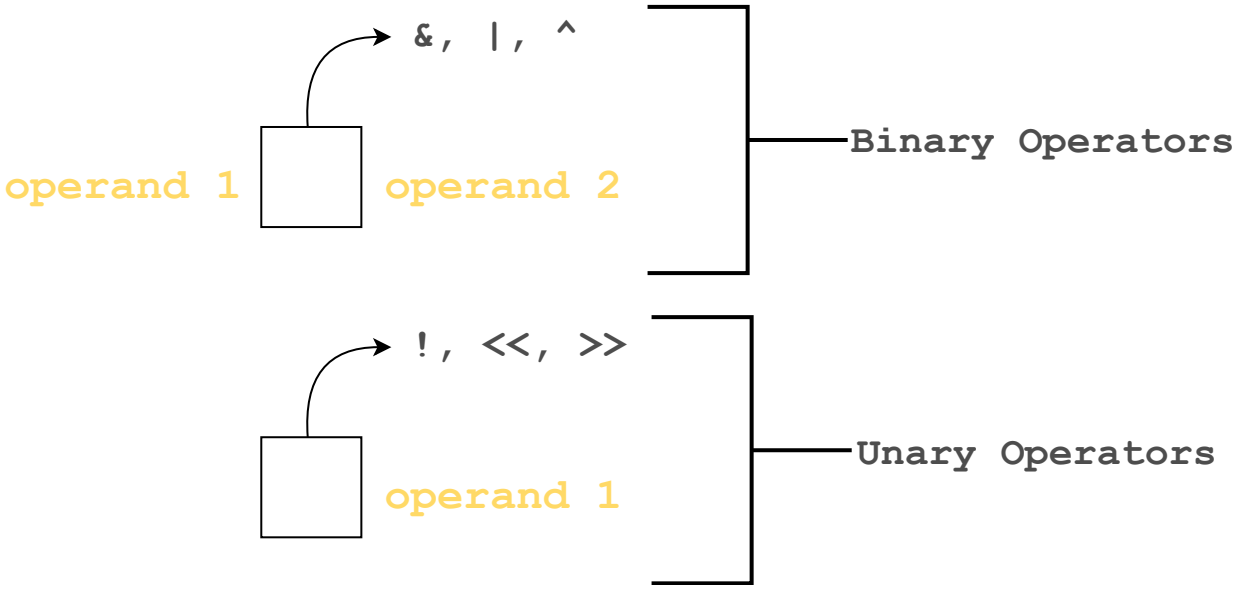
Bitwise Operators

This lesson discusses the bitwise operators in Rust.

We'll cover the following

What Are Bitwise Operators?

Bitwise operators deal with the binary representation of the operands.



Types

The table below summarizes the types of bitwise operators in Rust.

Operator	Operation	Explanation
operand 1 & operand 2	AND	Bitwise AND operand 1 and operand 2
operand 1 | operand 2	OR	Bitwise OR operand 1 and operand 2
operand 1 ^ operand 2	XOR	Bitwise XOR operand 1 and operand 2
! operand 1	NOT	Inverse the bits of the operand
<< operand 1	Left Shift	Moves all the bits in operand 1 to the left by the number of places specified in the operand 2. New bits are filled with zeros. Shifting a value left by one position is equivalent to multiplying it by 2, shifting two positions is equivalent to multiplying by 4, and so on.
>> operand 1	Right shift	Moves all the bits in operand 1 to the right by the number of places specified in the operand 2. New bits are filled with zeros.Shifting a value right by one position is equivalent to dividing it by 2, shifting two positions is equivalent to dividing by 4, and so on.

		dividing it by 2, shifting two positions is equivalent to dividing by 4, and so on.

Bitwise operators

> 📝 Note: Right shift `>>` is same as arithmetic right shift on signed integer types, logical right shift on unsigned integer types.

Example

The example below shows the bitwise AND, OR, XOR, Left Shift, and Right Shift operations.

operand 1 = 5 operand 2 = 6

operand 1 AND operand 2

```
  5 ( 1 0 1)
& 6 ( 1 1 0)
  ----------
( 1 0 0) => (4)
```

operand 1 OR operand 2

```
  5 ( 1 0 1)
| 6 ( 1 1 0)
  ----------
( 1 1 1) => (7)
```

operand 1 XOR operand 2

```
  5 ( 1 0 1)
^ 6 ( 1 1 0)
  ----------
( 0 1 1) => (3)
```

Left shift Add x zeros from the right

operand 1 << 2 → Left Shift two times. Fill the sifted places from right with zeros

```
<<( 0 0 0 0  0 1 0 1)
---------------------
  ( 0 0 0 1  0 1 0 0) => (20)
```

Right shift Add x zeros from the left

operand 1 >> 1 → Right Shift one time. Fill the sifted place from left with zero

```
>>(0 0 0 0 0 1 0 1)
-------------
  (0 0 0 0 0 0 1 0) => (2)
```

The following example shows the use of bitwise operators in a program:

```
fn main() {
  let a = 5;
  let b = 6;
```

```
  println!("Operand 1: {}, Operand 2: {}", a , b);
  println!("AND: {}", a & b);
  println!("OR: {}", a | b);

  println!("XOR: {}", a ^ b);
  println!("NOT a: {}", !a);
  println!("Left shift: {}", a << 2);
  println!("Right shift: {}", a >> 1);


}
```

Quiz

Test your understanding of bitwise operators in Rust.

Quick Quiz on Bitwise Operators!

Q

What is the output of the following code?

```
fn main() {
  let mut a = 1;
  let mut b = 2;
  a = a & b;
  a = a << 1;
  b = b >> 3;
  println!("a: {}", a);
  println!("b: {}", b);
}
```

Retake Quiz

The next lesson will cover the assignment operator in Rust.