

Precedence and Associativity

This lesson discusses two important characteristics of operators - precedence and associativity.

We'll cover the following



- Precedence
- Associativity
 - Left to Right Associativity
- Example 1
- Example 2
- Quiz

Precedence

The precedence of an operator determines which operation is performed first in an expression with more than one operators.

Operators are listed below in the order of their precedence from highest to lowest :

• Unary

- Logical/Bitwise NOT - `!`
- Dererence - `*`
- Borrow - `&`, `&mut`

• Binary

- Typecast - `as`
- Multiplication- `*`, Division - `/`, Remainder - `%`
- Addition - `+`, Subtraction - `-`
- Left Shift - `<<`, Right Shift - `>>`
- Bitwise AND - `&`
- Bitwise XOR - `^`
- Bitwise OR - `|`
- Comparison - `==`, `!=`, `<`, `>`, `<=`, `>=`

- Logical AND - `&&`
- Logical OR - `||`
- Range - `start .. stop`
- Assignment/Compound Assignment - `=` `+=` `-=` `*=` `/=` `%=` `&=` `|=` `^=` `<<=` `>>=`

Note: The operators that are written in the same row have the same order of precedence.

Note: The range operator will be further explored while we discuss the `for` loop in the [Loops](#) chapter.

Associativity


If two or more operators of the same precedence appear in a statement, then which operator will be evaluated first is defined by the associativity.

Left to Right Associativity

Left associativity occurs when an expression is evaluated from left to right. An expression such as `a ~ b ~ c`, in this case, would be interpreted as `(a ~ b) ~ c` where `~` can be any operator.

The operators below can be chained as left associative.

- `as`
- `*`, `/`, `%`
- `+`, `-`
- `<<` `>>`
- `&`
- `^`
- `|`
- `&&`
- `||`

 The comparison, assignment, and the range operator cannot be chained at all.

Example 1

The example below solves an expression according to its operator precedence:

```
fn main() {  
    println!("Answer : {}", ( 3 + 5 ) * 9 / 7 & 8);  
}
```



(3+5) * 9 / 7 & 8

8 * 9 / 7 & 8

$$(3+5) * 9 / 7 \& 8$$

$$8 * 9 / 7 \& 8$$

$$72 / 7 \& 8$$

2 of 4

$$(3+5) * 9 / 7 \& 8$$

$$8 * 9 / 7 \& 8$$

$$72 / 7 \& 8$$

$$10 \& 8$$

3 of 4

(3+5) * 9 / 7 & 8

8 * 9 / 7 & 8

72 / 7 & 8

10 & 8

8


4 of 4

—

[]

Example 2

The example below solves an expression according to its operator precedence:

 Solution3

```
fn test() {  
  println!("{}", 2 + 3 / 5 ^ 7 & 8 | 9);  
}
```



2 + 3 / 5 ^ 7 & 8 | 9

2 + 0 ^ 7 & 8 | 9

1 of 5

2 + 3 / 5 ^ 7 & 8 | 9

2 + 0 ^ 7 & 8 | 9

2 ^ 7 & 8 | 9

2 of 5

2 + 3 / 5 ^ 7 & 8 | 9

2 + 0 ^ 7 & 8 | 9

2 ^ 7 & 8 | 9

2 ^ 0 | 9

3 of 5

2 + 3 / 5 ^ 7 & 8 | 9

2 + 0 ^ 7 & 8 | 9

2 ^ 7 & 8 | 9

2 ^ 0 | 9

2 | 9

4 of 5

2 + 3 / 5 ^ 7 & 8 | 9

2 + 0 ^ 7 & 8 | 9

2 ^ 7 & 8 | 9

2 ^ 0 | 9

2 | 9

11

5 of 5

—

☐☐

Quiz

Test your understanding of operator precedence in Rust!

Quick Quiz on Operator Precedence!

Q

What is the output of the following code according to its operator precedence in Rust?

```
fn main() {  
    println!("{}", 3 + 4 - 9 / 6 * 6 ^ 8 & 3);  
}
```


[Retake Quiz](#)

Now that you have learned about the operator precedence, let's check your knowledge in the next lesson.