Operaciones

- Define alguna función (operación) que se realizará en los datos.
- Las operaciones se evaluan a un solo valor, luego se enlaza a una variable
- Ejemplo:
  - `5 + 4`
  - Donde
    - `5` y `4` son operandos
    - `+` es el operador

Aritmeticas

- Son utilizados para sumar, restar, multiplicar, dividir y obtener el residuo entre **dos operandos**
- Retornan un numero del mismo tipo de dato que los operandos

Adición +

```
let a=23;
let b= 7;
println!("suma: {}",a+b);
```

- Devuelve la suma de los operandos `a + b`.

Sustracción -

```
let a=23;
let b= 7;
println!("resta: {}",a-b);
```

- Devuelve la diferencia de los operandos `a - b`.

Multiplicación *

```
let a=23;
let b= 7;
println!("multiplicación: {}",a*b);
```

- Devuelve el producto de los operandos `a * b`.

División /

```
let a=23;
let b= 7;
println!("división: {}",a/b);
```

- Devuelve el cociente de la división `a / b`.

Modulo %

```
let a=23;
let b= 7;
println!("modulo: {}",a%b);
```

- Devuelve el resto de la división `a % b`

Relacionales

- Son utilizados para comparar valores
- Retornar un valor booleano: **true y false**

Mayor que >

```
let a=6;
let b=4;
println!("mayor que: {}",a>b);
```

- Devuelve la veracidad de la premisa `a > b`

Menor que <

```
let a=6;
let b=4;
println!("menor que: {}",a<b);
```

- Devuelve la veracidad de la premisa `a < b`

Mayor o igual que >=

```
let a=6;
let b=4;
println!("mayor o igual que: {}",a>=b);
```

- Devuelve la veracidad de la premisa `a >= b`

Menor o igual que <=

```
let a=6;
let b=4;
println!("menor o igual que: {}",a<=b);
```

- Devuelve la veracidad de la premisa `a <= b`

Igual que ==

```
let a=6;
let b=4;
println!("igual: {}",a==b);
```

- Devuelve la veracidad de la premisa `a == b`

Diferente !=

```
let a=6;
let b=4;
println!("diferente: {}",a!=b);
```

- Devuelve la veracidad de la premisa `a != b`

Lógicas

- Utilizados para combinar más de una condición
- Retornar un valor booleano: **true o false**

AND &&

```
let a=true;
let b=false;
println!("and: {}", a && b);
```

- Devuelve verdadero si todas las expresiones retornan verdadero

OR ||

```
let a=true;
let b=false;
println!("or: {}", a && b);
```

- Devuelve verdadero si al menos una expresión retorna verdadero

Negación !

```
let a=true;
let b=false;
println!("no: {}",!b);
```

- Devuelve la negación del valor booleano

Bit a bit

AND en bits &

```
let a=4;
let b=2;
println!("and en bits: {}",a&b);
```

- Realiza la operación AND en cada bit de los enteros.

OR en bits |

```
let a=4;
let b=2;
println!("or en bits: {}",a|b);
```

- Realiza la operación OR en cada bit de los enteros.

XOR en bits ^

```
let a=4;
let b=2;
println!("xor en bits: {}",a^b);
```

- Realiza la operación XOR (OR exclusivo, ambos no pueden ser verdaderos) en cada bit de los enteros.

Invertir bits !

```
let b=2;
println!("invertir bits: {}",!b);
```

- Invierte todos los bits en el operadorando.

Desplazar bits a la izquierda <<

```
let a=4;
println!("desplazar bits a la izquierda: {}",a<<1);
```

- Mueve todos los bits del primer operando a la izquierda por el número de lugares especificados en el segundo operando, los nuevos bits se rellenan con ceros. (*Cambiar un valor a la izquierda es multiplicarlo por 2*).

Desplazar bits a la derecha >>

```
let a=4;
println!("desplazar bits a la derecha: {}",a>>1);
```

- Mueve todos los bits del primer operando a la derecha por el número de lugares especificados en el segundo operando.