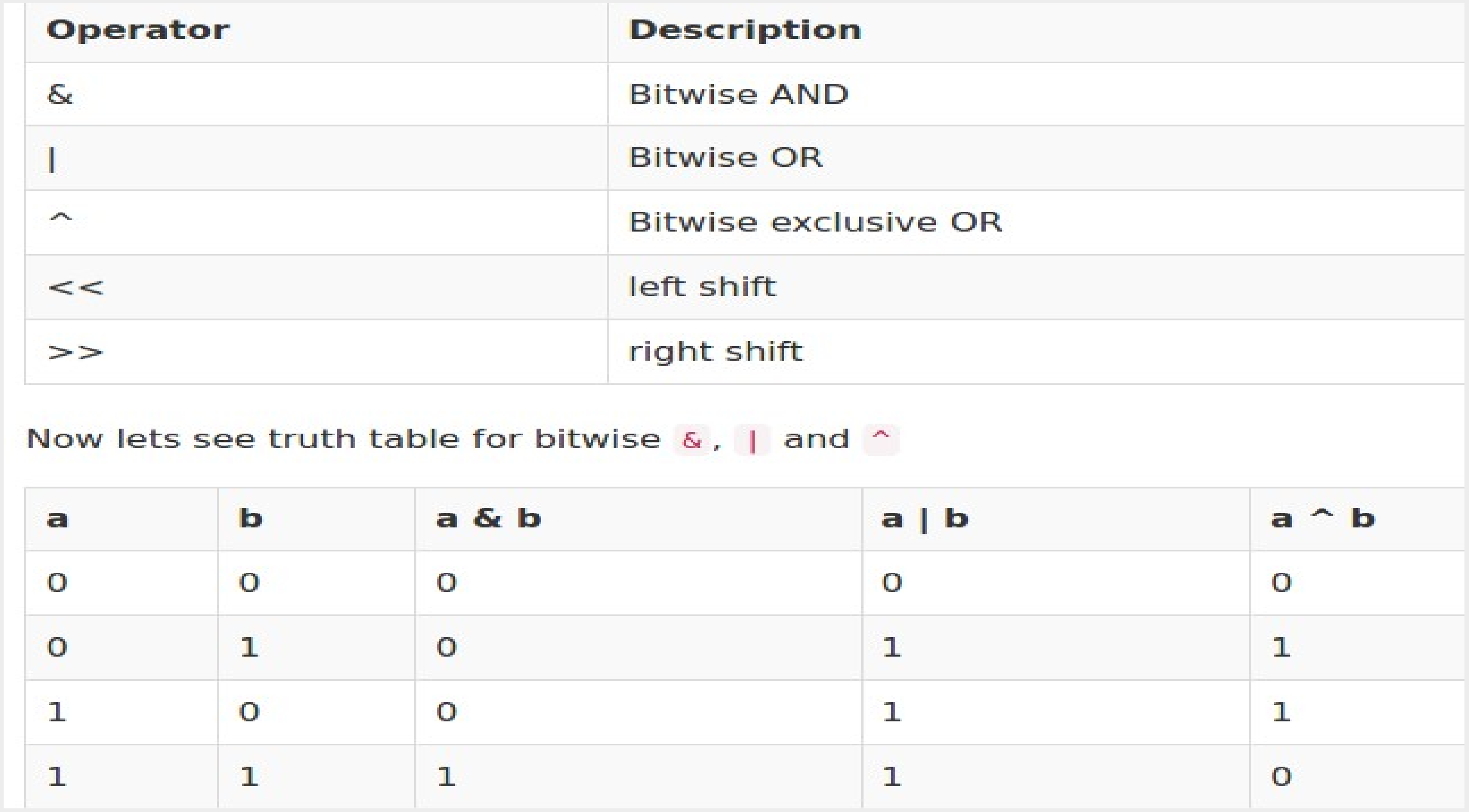
Operators

- **Bitwise Operators:** Bitwise operators perform manipulations of data at bit level. These operators also perform shifting of bits from right to left.
- Bitwise operators are not applied to float or double.



Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	left shift
>>	right shift

Now lets see truth table for bitwise `&`, `|` and `^`

a	b	a & b	a | b	a ^ b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Operators

- **Bitwise AND operator &:** The output of bitwise AND is 1 if the corresponding bits of two operands is 1. If either bit of an operand is 0, the result of corresponding bit is evaluated to 0.
- Let us suppose the bitwise AND operation of two integers 12 and 25:

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bit Operation of 12 and 25

```
  00001100
& 00011001
  00001000  = 8 (In decimal)
```

Operators

- **Bitwise OR operator | :** The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1. In C Programming, bitwise OR operator is denoted by |.

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bitwise OR Operation of 12 and 25

```
  00001100
| 00011001
  ________
  00011101  = 29 (In decimal)
```

Operators

- **Bitwise XOR (exclusive OR) operator ^ :** The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite. It is denoted by ^.

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bitwise XOR Operation of 12 and 25

```
  00001100
^ 00011001
  ________
  00010101  = 21 (In decimal)
```

Operators

- **Bitwise Operators:** The bitwise shift operator, shifts the bit value. The left operand specifies the value to be shifted and the right operand specifies the number of positions that the bits in the value have to be shifted. Both operands have the same precedence.

Operators

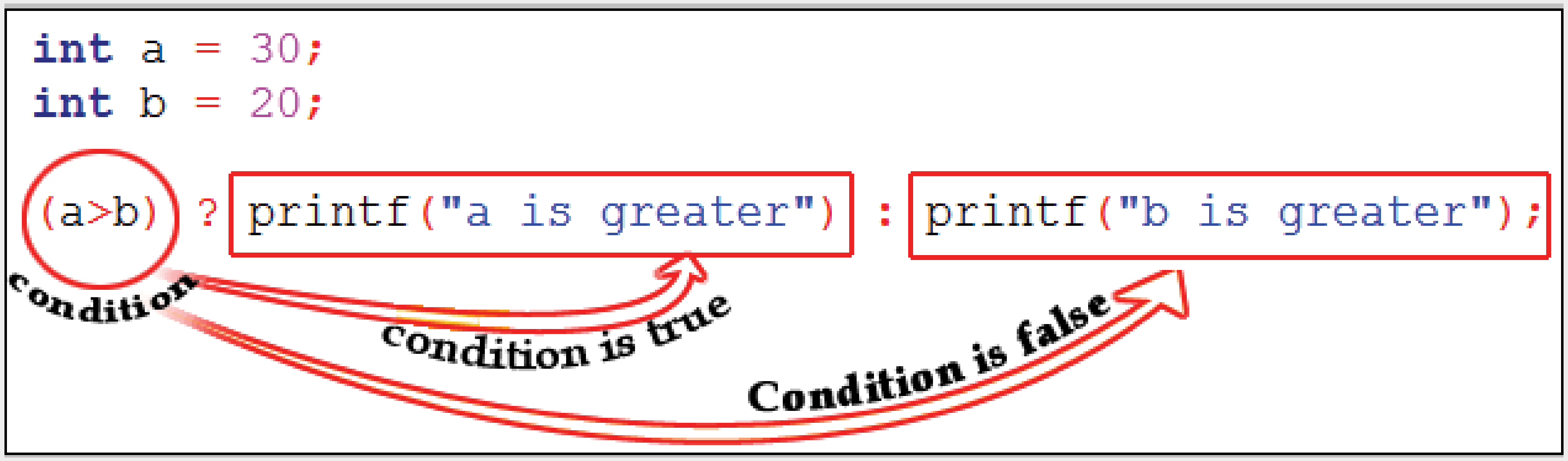
- **Bitwise complement operator ~ :** Bitwise complement of any number N is -(N+1)
- **Shift Operators in C programming**
- There are two shift operators in C programming:
  - Right shift operator
  - Left shift operator.
- **Right Shift Operator**
- Right shift operator shifts all bits towards right by certain number of specified bits. It is denoted by >>.
  - 212 = 11010100 (In binary)
  - 212>>2 = 00110101 (In binary) [Right shift by two bits]
  - 212>>7 = 00000001 (In binary)
  - 212>>8 = 00000000
  - 212>>0 = 11010100 (No Shift)

Operators

- **Left Shift Operator:**
- Left shift operator shifts all bits towards left by a certain number of specified bits. The bit positions that have been vacated by the left shift operator are filled with 0. The symbol of the left shift operator is <<.
  - 212 = 11010100 (In binary)
  - 212<<1 = 110101000 (In binary) [Left shift by one bit]
  - 212<<0 = 11010100 (Shift by 0)
  - 212<<4 = 110101000000 (In binary) =3392(In decimal)

Operators

- **Conditional Operators:** The conditional operators in C language are known by two more names
  - **Ternary Operator**
  - **? : Operator**
- It is actually the if condition that we use in C language decision making, but using conditional operator, we turn the if condition statement into a short and simple operator.
- The syntax of a conditional operator is :
  - expression 1 ? expression 2: expression 3

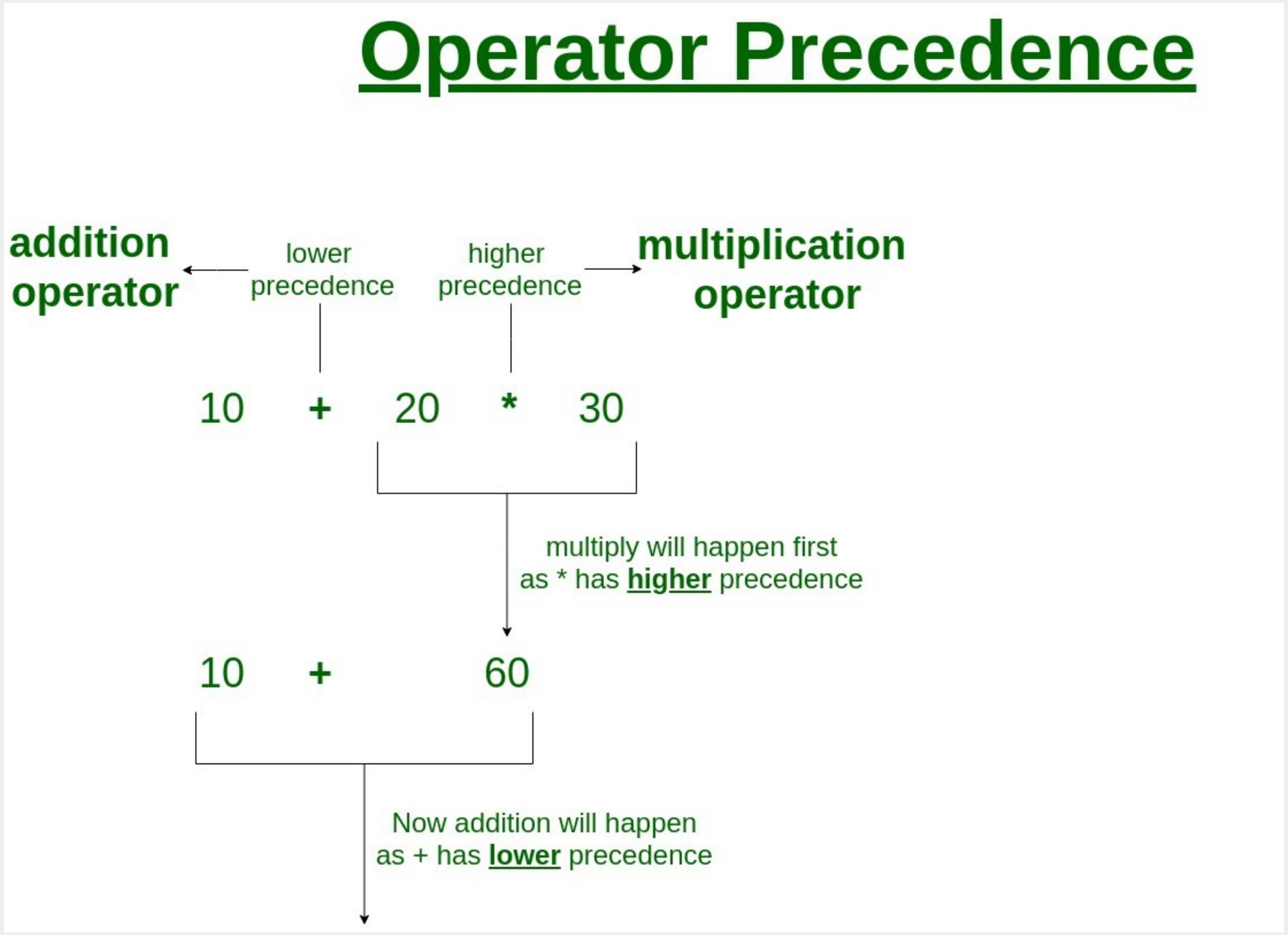


```
int a = 30;
int b = 20;

(a>b) ? printf("a is greater") : printf("b is greater");
```

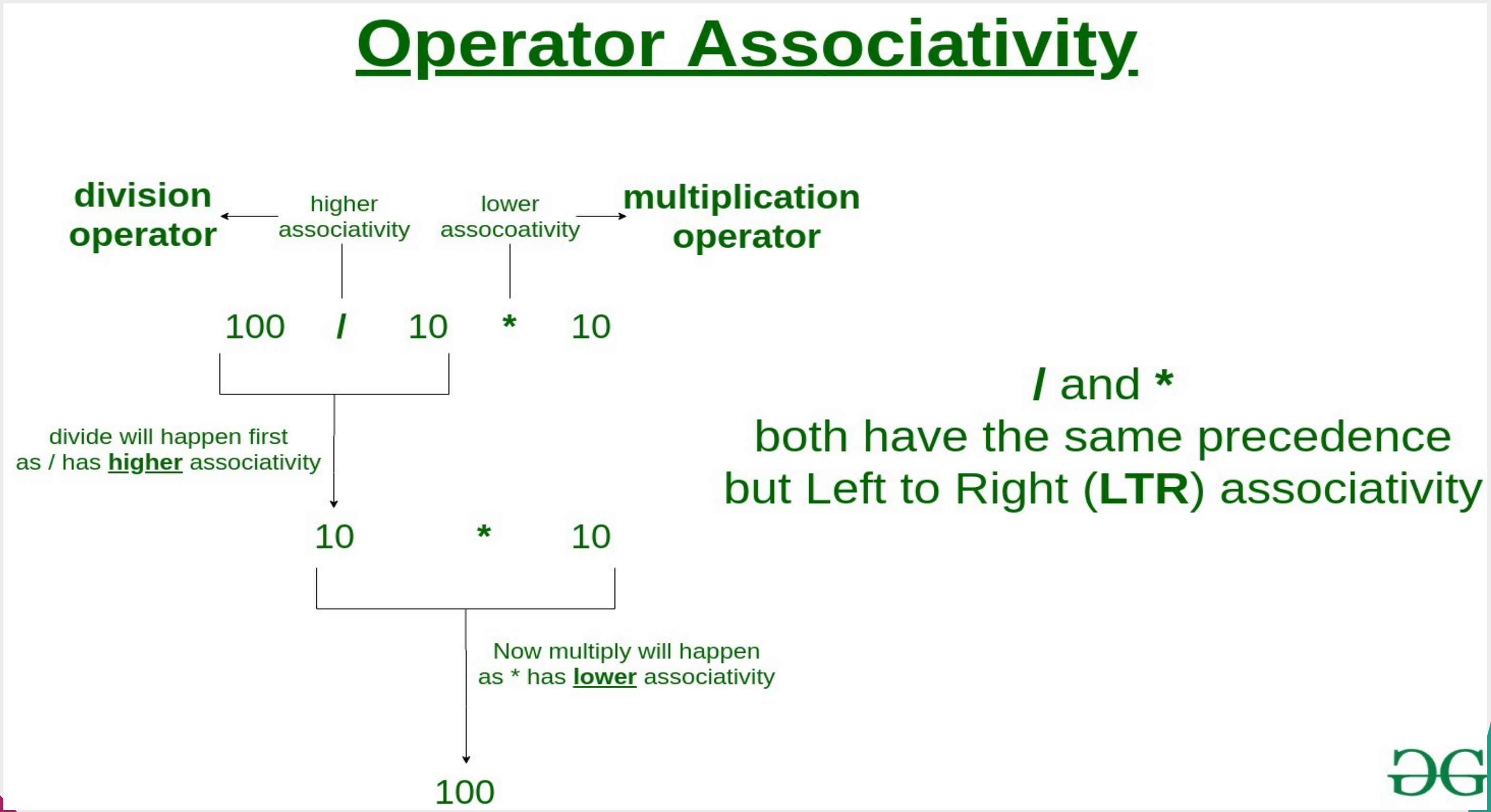
Operator Precedence and Associativity in C

- **Operator precedence determines which operator is performed first in an expression with more than one operators with different precedence.**
- **For example: Solve - 10 + 20 * 30**



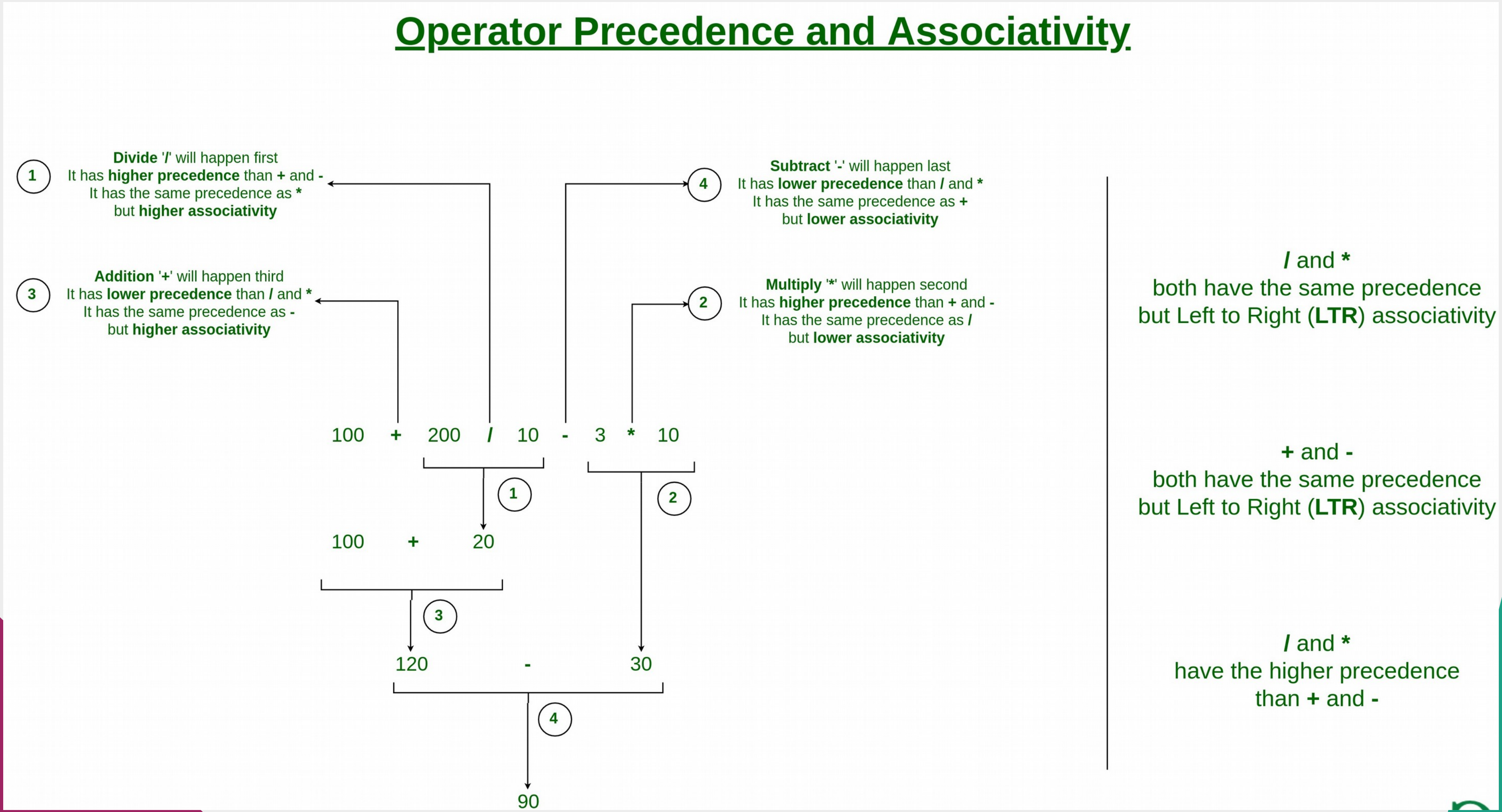
Operator Precedence and Associativity in C

- 10 + 20 * 30 is calculated as 10 + (20 * 30) and not as (10 + 20) * 30
- Operators Associativity is used when two operators of same precedence appear in an expression. Associativity can be either Left to Right or Right to Left.
- For example: ‘*’ and ‘/’ have same precedence and their associativity is Left to Right, so the expression “100 / 10 * 10” is treated as “(100 / 10) * 10”.

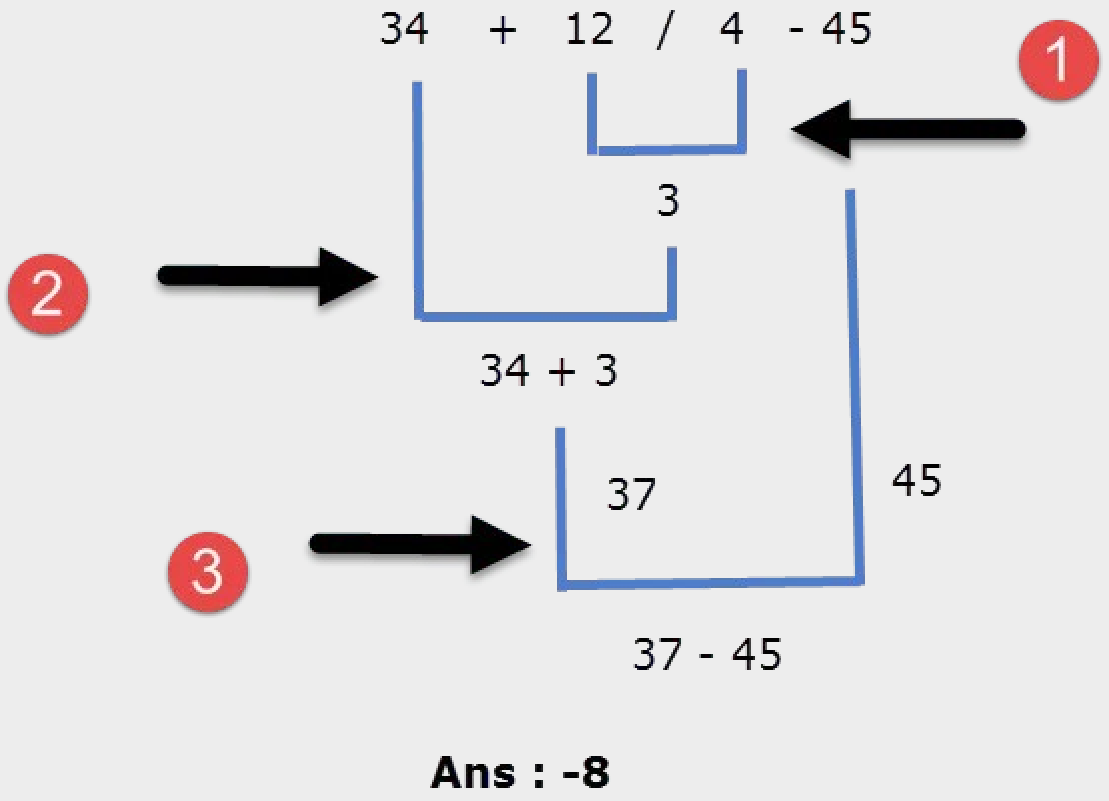

Operator Associativity
division operator
higher associativity
lower assocoativity
multiplication operator
100 / 10 * 10
divide will happen first as / has higher associativity
10 * 10
Now multiply will happen as * has lower associativity
100
/ and * both have the same precedence but Left to Right (LTR) associativity

Operator Precedence and Associativity in C

- Operators Precedence and Associativity are two characteristics of operators that determine the evaluation order of sub-expressions in absence of brackets.
- **For example: Solve - 100 + 200 / 10 - 3 * 10**


Operator Precedence and Associativity
1 Divide '/' will happen first
It has higher precedence than + and -
It has the same precedence as *
but higher associativity
3 Addition '+' will happen third
It has lower precedence than / and *
It has the same precedence as -
but higher associativity
4 Subtract '-' will happen last
It has lower precedence than / and *
It has the same precedence as +
but lower associativity
2 Multiply '*' will happen second
It has higher precedence than + and -
It has the same precedence as /
but lower associativity
100 + 200 / 10 - 3 * 10
100 + 20
120 - 30
90
/ and * both have the same precedence but Left to Right (LTR) associativity
+ and - both have the same precedence but Left to Right (LTR) associativity
/ and * have the higher precedence than + and -

Operator Precedence and Associativity in C



34 + 12 / 4 - 45
1
3
2
34 + 3
37
45
3
37 - 45
Ans : -8