Lecture-5(Part-2)Operators

Precedence and Associativity of Operators

Precedence of Operators

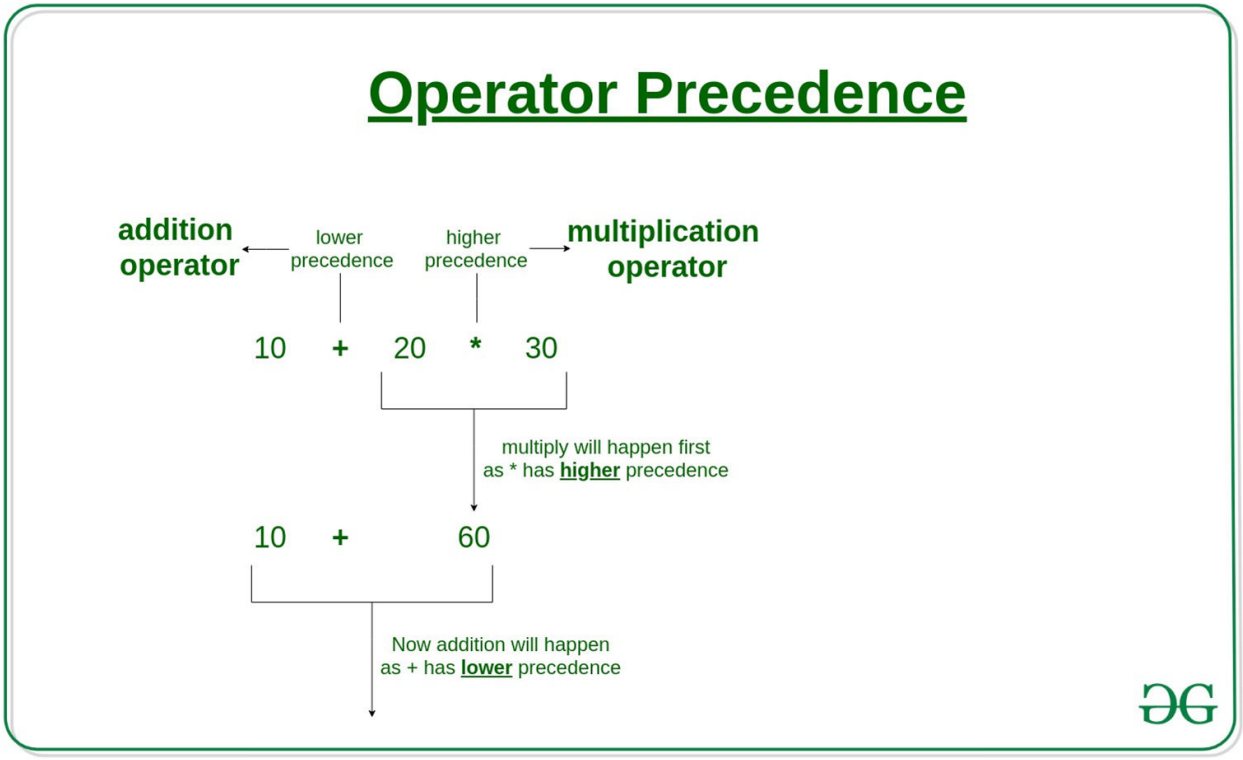
- The precedence of operators determine a rank for the operators. The higher an operator's precedence or priority,

Example: So how the expression a * b + c will be interpreted?

(a * b) + c or a * (b + c),

here the first interpretation is the one that is used because the multiplication operator has higher precedence than addition.

Precedence-Example



Associativity of Operators

- Associativity tell us the order in which several operators with equal precedence are computed or processed in two directions, either from left to right or vice-versa.

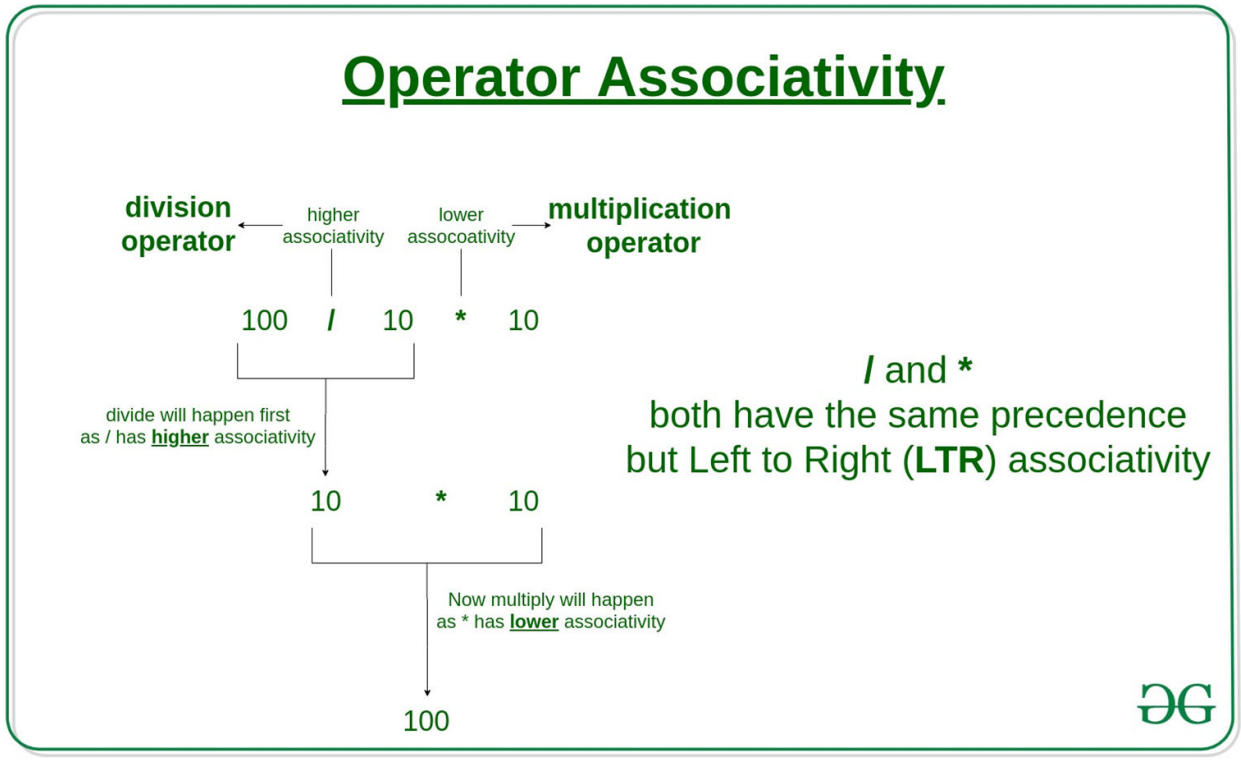
Example: In the expression

a * b / c,

since multiplication and division have the same precedence we must use the associativity to determine the grouping. These operators are left associative which means they are grouped left to right as if the expression was

(a * b) / c.

Associativity-Example



Operator	Description	Associativity
( ) [ ] . -> ++ --	Parentheses (function call) (see Note 1) Brackets (array subscript) Member selection via object name Member selection via pointer Postfix increment/decrement (see Note 2)	left-to-right
++ -- + - ! ~ (*type*) * & sizeof	Prefix increment/decrement Unary plus/minus Logical negation/bitwise complement Cast (convert value to temporary value of *type*) Dereference Address (of operand) Determine size in bytes on this implementation	right-to-left
* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right
< <= > >=	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right
== !=	Relational is equal to/is not equal to	left-to-right
&	Bitwise AND	left-to-right
^	Bitwise exclusive OR	left-to-right
|	Bitwise inclusive OR	left-to-right
&&	Logical AND	left-to-right
||	Logical OR	left-to-right
? :	Ternary conditional	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Addition/subtraction assignment Multiplication/division assignment Modulus/bitwise AND assignment Bitwise exclusive/inclusive OR assignment Bitwise shift left/right assignment	right-to-left
,	Comma (separate expressions)	left-to-right

Example Expressions Using the Precedence Chart

```
1. x = 3 * 4 + 5 * 6
     = 12 + 5 * 6
     = 12 + 30
     = 42
2. x = 3 * (4 + 5) * 6
     = 3 * 9 * 6
     = 27 * 6
     = 162
3. x = 3 * 4 % 5 / 2
     = 12 % 5 / 2
     = 2 / 2
     = 1
4. x = 3 * (4 % 5) / 2
     = 3 * 4 / 2
     = 12 / 2
     = 6
5. x = 3 * 4 % (5 / 2)
     = 3 * 4 % 2
     = 12 % 2
     = 0
6. x = 3 * ((4 % 5) / 2)
     = 3 * (4 / 2)
     = 3 * 2
     = 6
```

Consider

int a=0,b=1,c=-1

float x=2.5,y=0.0

```
10. a && b
    = 0
11. a < b && c < b
    = 1
12. b + c || ! a
    = ( b + c) || (!a)
    = 0 ||1
    = 1
13. x * 5 && 5 || ( b / c)
    = ((x * 5) && 5) || (b / c)
    = (12.5 && 5) || (1/-1)
    = 1
14. a <= 10 && x >= 1 && b
    = ((a <= 10) && (x >= 1)) && b
    = (1 && 1) && 1
    = 1
15. !x || !c || b + c
    = ((!x) || (!c)) || (b + c)
    = (0 || 0) || 0
    = 0
```

Consider

int a=0,b=1,c=-1

float x=2.5,y=0.0

Practice questions(Q1)

```
#include <stdio.h>
 int main()
  {
     double b = 5 % 3 & 4 + 5 * 6;
     printf("%lf", b);
     return 0;
  }
```

A. 2
B. 30
C. 2.000000
D. 5

(Q2)

```
#include <stdio.h>
   int main()
   {
      double b = 3 && 5 & 4 % 3;
      printf("%lf", b);
      return 0;
   }
```

A. 3.000000
B. 4.000000
C. 5.000000
D. 1.000000

(Q3)

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    double b = 5 & 3 && 4 || 5 | 6;
    printf("%lf", b);
    return 0;
}
```

A. 1.000000

B. 0.000000

C. 7.000000

D. 2.000000

(Q4)

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int b = 5 + 7 * 4 - 9 * (3, 2);
    printf("%d", b);
}
```

A. 6

B. 15

C. 13

D. 21

(Q5)

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int b = 4 * 6 + 3 * 4 < 3 ? 4 : 3;
    printf("%d\n", b);
}
```

A. 3

B. 33

C. 34

D. 4

(Q6)

Which of the following operators has an associativity from Right to Left?

A. <=

B. <<

C. ==

D. +=

(Q7)

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int a = -1, b = 4, c = 1, d;
    d = ++a && ++b || ++c;
    printf("%d, %d, %d, %d\n", a, b, c, d);
    return 0;
}
```

A. 0, 4, 2, 1

B. 0, 5, 2, 1

C. -1, 4, 1, 1

D. 0, 5, 1, 0