Operators and Expressions

Dr. Nachiket Tapas

Review

```
main()
{
        int a, b, c, d;
        a = 15;
        b = 10;
        c = ++a - b;
        printf(“a = %d b = %d c = %d\n”,a, b, c);
        d = b++ +a;
        printf(“a = %d b = %d d = %d\n”,a, b, d);
        printf(“a/b = %d\n”, a/b);
        printf(“a%%b = %d\n”, a%b);
        printf(“a *= b = %d\n”, a*=b);
        printf(“%d\n”, (c>d) ? 1 : 0);
        printf(“%d\n”, (c<d) ? 1 : 0);
}
```

Output

a = 16 b = 10 c = 6
a = 16 b = 11 d = 26
a/b = 1
a%b = 5
a *=b = 176
0
1

Review

```
main()
{
        float a, b, c, x, y, z;
        a = 9;
        b = 12;
        c = 3;
        x = a – b / 3 + c * 2 - 1;
        y = a – b / (3 + c) * (2 - 1);
        z = a – (b / (3 + c) * 2) - 1;
        printf(“x = %f\n”, x);
        printf(“y = %f\n”, y);
        printf(“z = %f\n”, z);
}
```

Output

x = 10.000000
y = 7.000000
z = 4.000000

Is the following true?

- !(5 + 5 >=10)
- 5 > 10 || 10 < 20 && 3 < 5
- 10 ! = 15 && !(10<20) || 15 > 30

Types of operators

- Arithmetic operators
- Relational operators
- Logical operators
- Increment and decrement operators
- Assignment operators
- Conditional operators
- Bitwise operators
- Special operators

Bitwise Operators

&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Shift Left
>>	Shift Right

Example

```
#include <stdio.h>
int main() {
        int a = 5, b = 9;
        printf("%d\n", a&b);
        printf("%d\n", a|b);
        printf("%d\n", a^b);
        printf("%d\n", a<<1);
        printf("%d\n", a>>1);
        return 0;
}
```

Special Operators

- comma operator
  - value = (x=5, y=7, x+y);
  - printf(“%d”, value);

- sizeof()
  - printf(“%d”, sizeof(int));

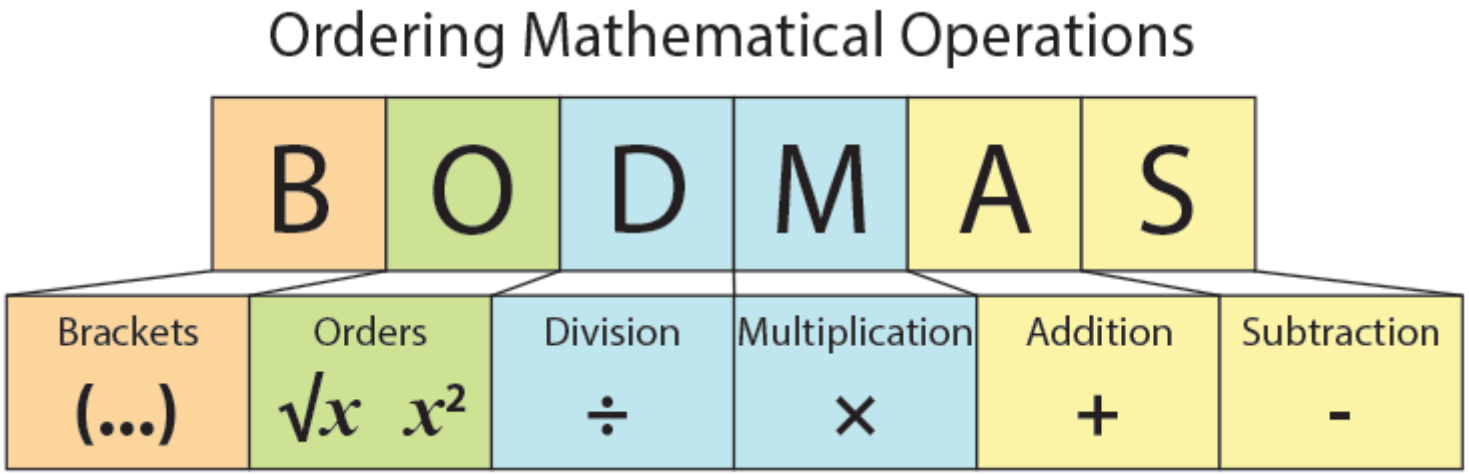
What is the result?

- [7 + 8 { 4 × (6 + 4 × 3) × 4}]

What about this?

- $45 + 3 \{ 34 – 18 – 14 \} \div 3 [17 + 3 \times 4 – (2 \times 7)]$

Operator Precedence



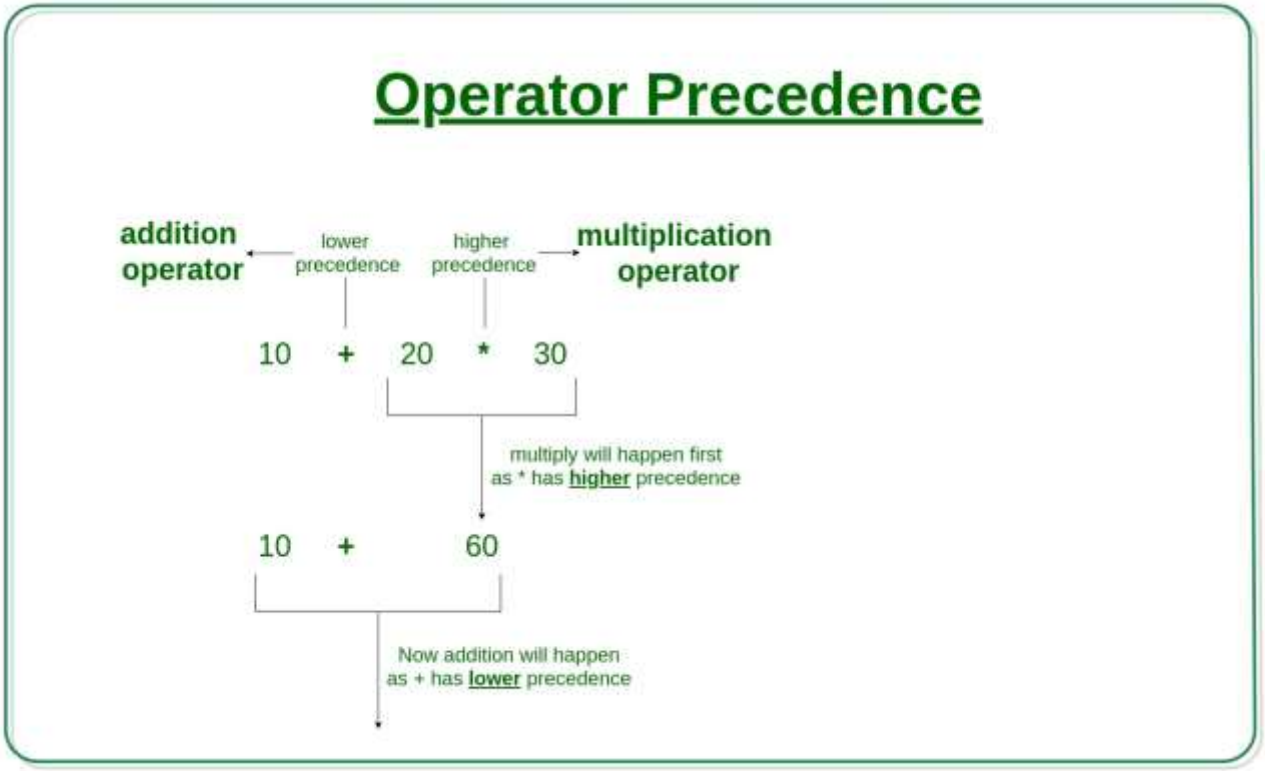
Operator Precedence in C

Precedence	Operator	Description	Associativity
1	++ -- () [] . -> (*type*){*list*}	Suffix/postfix increment and decrement Function call Array subscripting Structure and union member access Structure and union member access through pointer Compound literal(C99)	Left-to-right
2	++ -- + - ! ~ (*type*) * & sizeof _Alignof	Prefix increment and decrement[note 1] Unary plus and minus Logical NOT and bitwise NOT Cast Indirection (dereference) Address-of Size-of[note 2] Alignment requirement(C11)	Right-to-left
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	
5	<< >>	Bitwise left shift and right shift	
6	< <= > >=	For relational operators < and ≤ respectively For relational operators > and ≥ respectively	
7	== !=	For relational = and ≠ respectively	
8	&	Bitwise AND	
9	^	Bitwise XOR (exclusive or)	
10	|	Bitwise OR (inclusive or)	
11	&&	Logical AND	
12	||	Logical OR	
13	?:	Ternary conditional[note 3]	Right-to-left
14[note 4]	= += -= *= /= %= <<= >>= &= ^= |=	Simple assignment Assignment by sum and difference Assignment by product, quotient, and remainder Assignment by bitwise left shift and right shift Assignment by bitwise AND, XOR, and OR	
15	,	Comma	Left-to-right

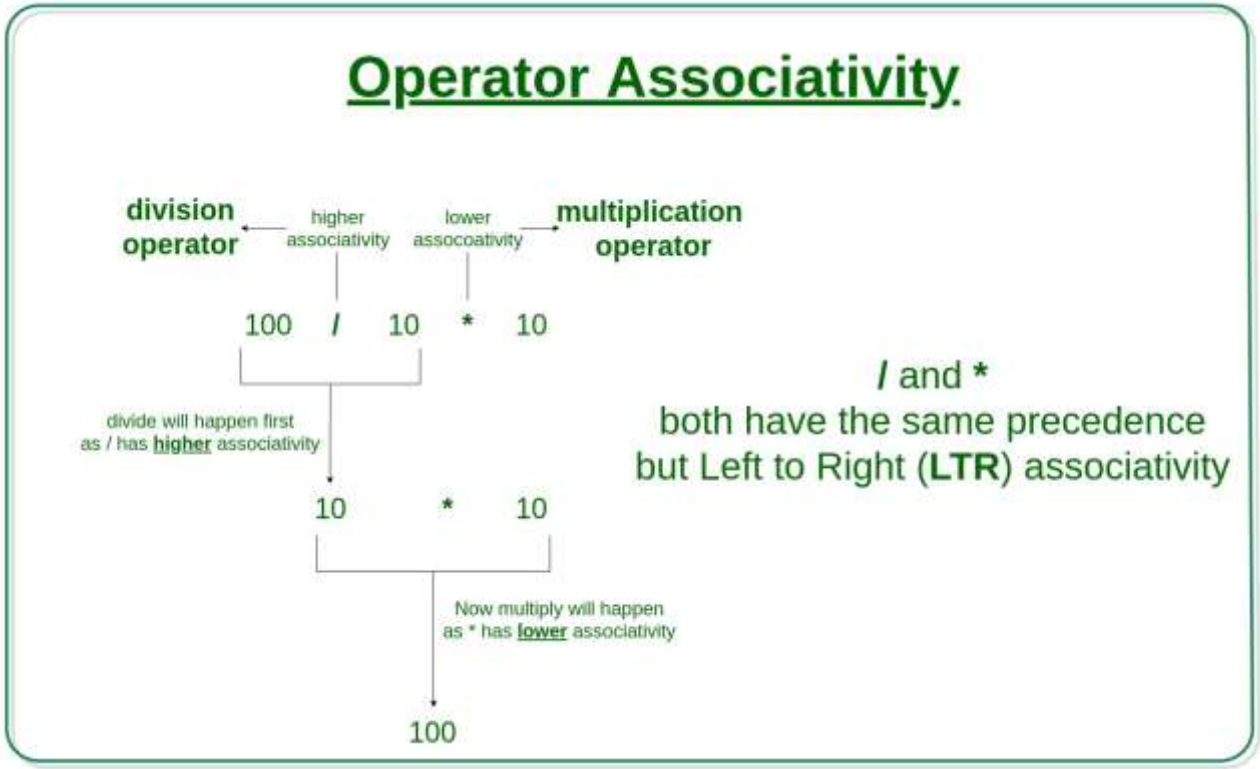
When to use associativity?

- **We only use associativity when we have two or more operators that have the same precedence in an expression.**
- **The point to note here is that associativity is not applicable when we are defining the order of evaluation of operands with different levels of precedence.**

Example


Operator Precedence
addition operator
lower precedence
higher precedence
multiplication operator
10 + 20 * 30
multiply will happen first as * has higher precedence
10 + 60
Now addition will happen as + has lower precedence

Example


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

Example

- 12 + 3 – 4 / 2 < 3 + 1

Code

```
#include <stdio.h>
main() {
   int a = 20;
   int b = 10;
   int c = 15;
   int d = 5;
   int e;
   e = (a + b) * c / d; // ( 30 * 15 ) / 5
   printf("Value of (a + b) * c / d is : %d\n", e );
   e = ((a + b) * c) / d; // (30 * 15 ) / 5
   printf("Value of ((a + b) * c) / d is : %d\n" , e );
   e = (a + b) * (c / d); // (30) * (15/5)
   printf("Value of (a + b) * (c / d) is : %d\n", e );
   e = a + (b * c) / d; // 20 + (150/5)
   printf("Value of a + (b * c) / d is : %d\n" , e );
   return 0;
}
```

```
Value of (a + b) * c / d is : 90
Value of ((a + b) * c) / d is : 90
Value of (a + b) * (c / d) is : 90
Value of a + (b * c) / d is : 50
```

Is the following true?

- 5 + 5 = = 10 || 1 + 3 = = 5