Lecture - 11 ✓

Programming in C ✓

Recalling Statement, Expression & Equation

⇒ Statements : An instructions. < declaration `int x;` / definition `int x = 10;`

↳ Every line of code is a Statement

↳ `# include < stdio.h >`

$\underbrace{\texttt{printf("\%d", 10)}}$

Print Statement

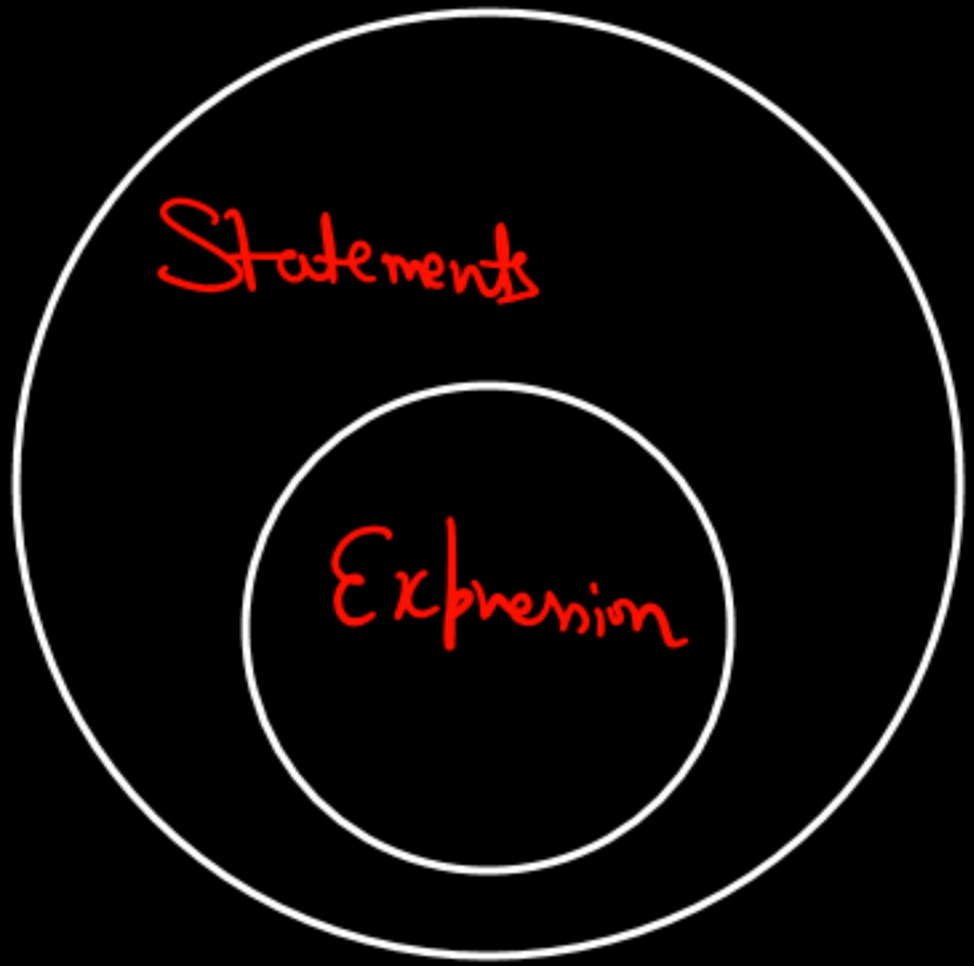
⇒ Expression → The type of Statements having a value. →

```
int m;    ← Statement
m = 24;   ← Expression
```

⇒ Equation → LHS = RHS

↳
```
int a=20;  int b=20;
a+b = 40;
```
↓ ← NOT Allowed

Error



Operator Precedence :

↳ which operator has to solve first.

Ex

$\underbrace{10 + 10}_{\downarrow} \times 2$ + ↓

20×2 *

$= 40$

$10 + \underbrace{10 \times 2}$ * ↓

$10 + 20$ +

$= 30$ ✓

Ex

$\underbrace{10 / 2}_{\downarrow} * 3$ ↓ ÷

5×3

$\underline{\underline{15}}$ ✓ ×

$10 / \underbrace{2 * 3}$ * ↓

$10 / 6$ /

$0.\overline{666}$

Grouping
↓
Some operator have same Precedence
↓
then we Check for Associativity
↓
tells about direction
⟵
Left to Right
Right to Left

Precedence of Operator

1) Postfix operator (x++, x--) (Left to Right)

```
int a = 1 ;

           a=2        a=1
int b = a++  +  a--  +  a++  ;
  ↓      ↓       ↓       ↓
  4      1   +   2   +   1   =  4
        a=2     a=1     a=2
```

2) Unary Operator → [Right to Left]

↳ ++x, --x

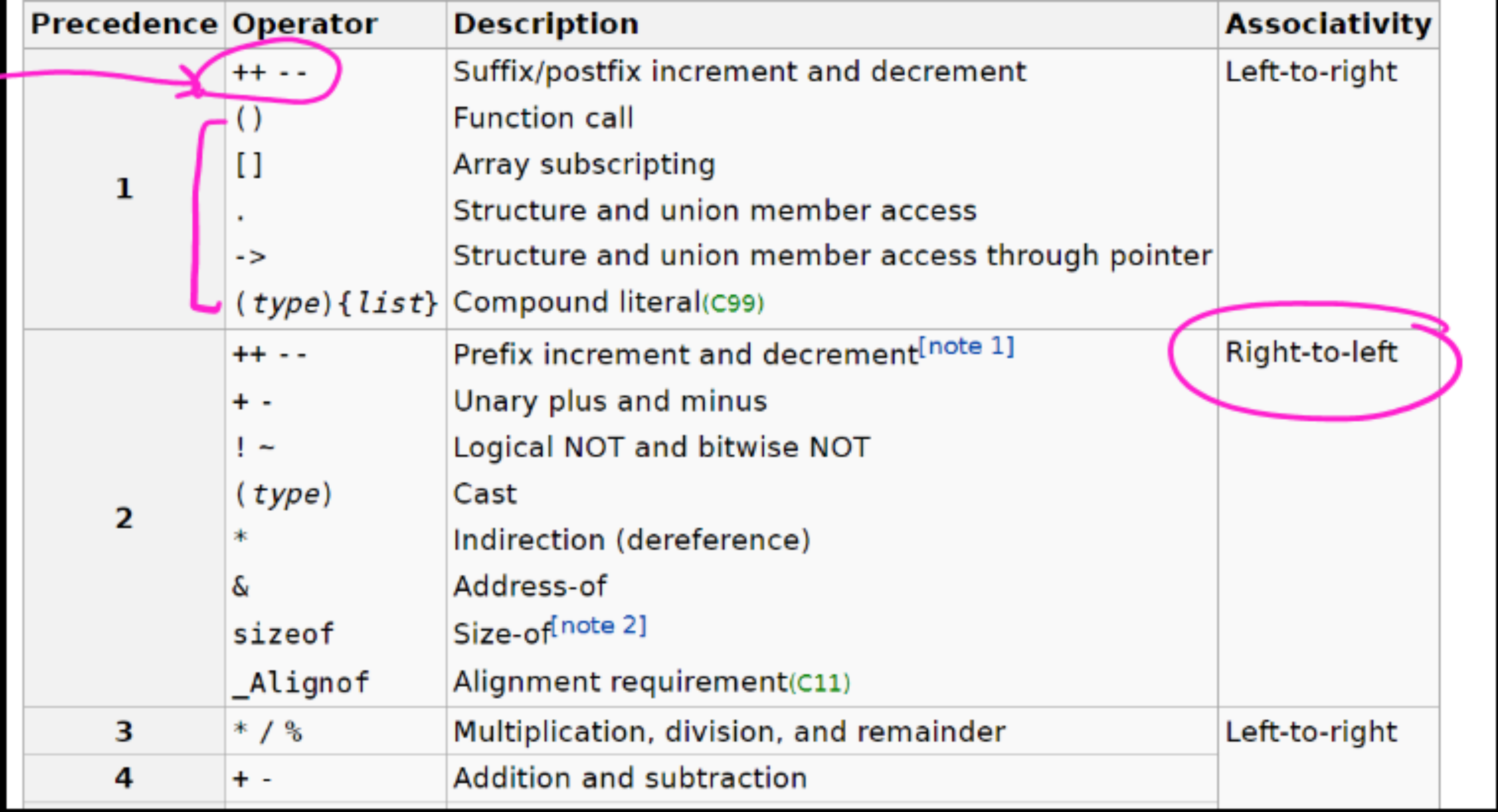
→ +, - (Unary)

→ Logical NOT & Bitwise NOT
  ⇓ ! ~

→ Address &

→ typecasting operator → (int) 3.14

→ Size of ( ) ←



Precedence	Operator	Description	Associativity
1	++ --	Suffix/postfix increment and decrement	Left-to-right
	( )	Function call	
	[ ]	Array subscripting	
	.	Structure and union member access	
	->	Structure and union member access through pointer	
	(*type*){*list*}	Compound literal(C99)	
2	++ --	Prefix increment and decrement[note 1]	Right-to-left
	+ -	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(*type*)	Cast	
	*	Indirection (dereference)	
	&	Address-of	
	sizeof	Size-of[note 2]	
	_Alignof	Alignment requirement(C11)	
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	

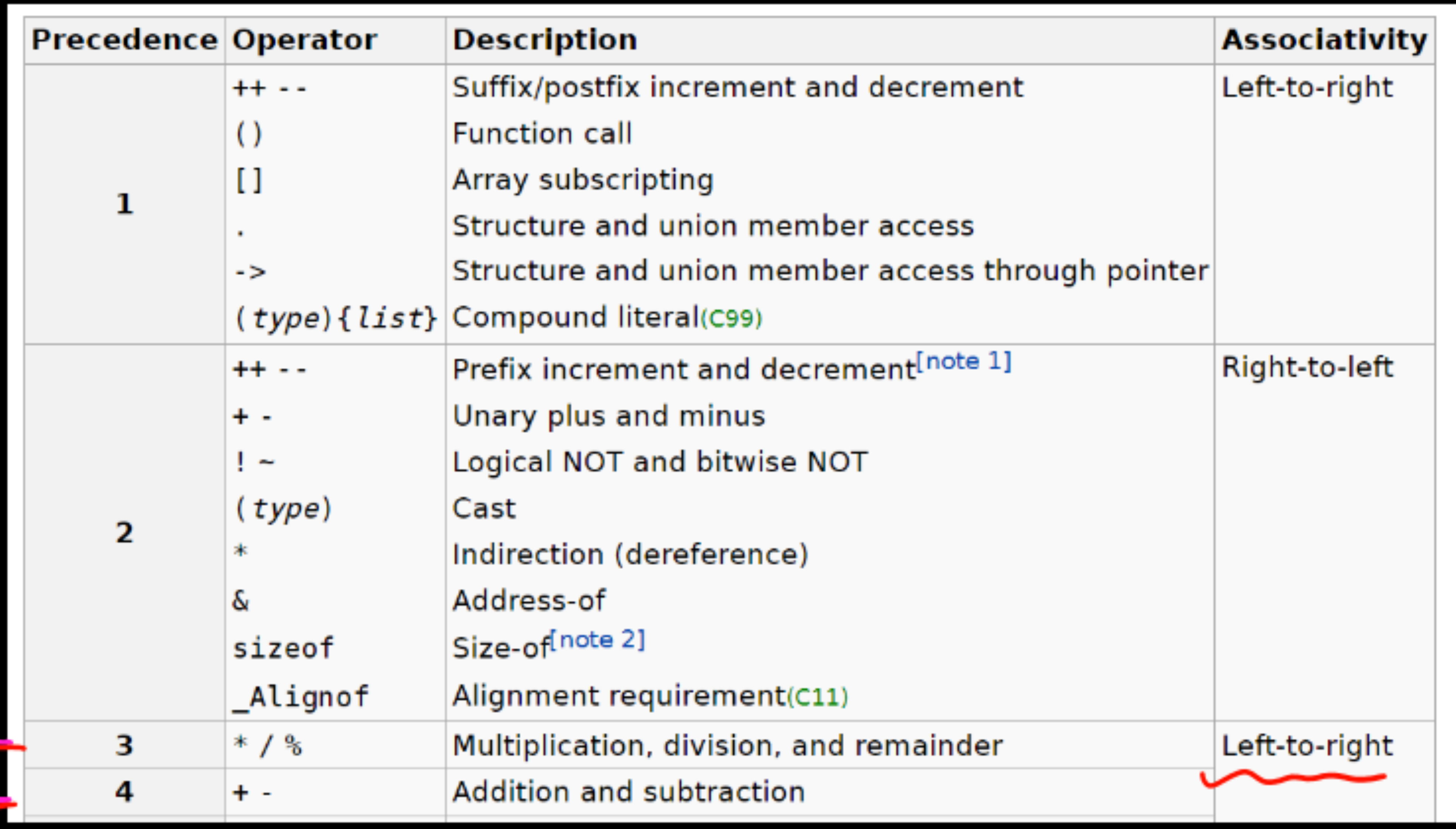
Ex int a = 10;

$$\text{int } b = \underset{\downarrow}{\text{sizeof}(a)} + \underset{\downarrow}{\underbrace{++a}} + \underset{\downarrow}{\underbrace{(\text{float})\, a}}$$

$$4 + 11.0 + 10.0$$

int (25.0)
↳ 25

sizeof, ++a, (float) ← Same Precedence
↳ Associativity → R→L



Precedence	Operator	Description	Associativity
1	++ --	Suffix/postfix increment and decrement	Left-to-right
	( )	Function call	
	[ ]	Array subscripting	
	.	Structure and union member access	
	->	Structure and union member access through pointer	
	(*type*){*list*}	Compound literal(C99)	
2	++ --	Prefix increment and decrement[note 1]	Right-to-left
	+ -	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(*type*)	Cast	
	*	Indirection (dereference)	
	&	Address-of	
	sizeof	Size-of[note 2]	
	_Alignof	Alignment requirement(C11)	
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	

3) Arithmetic Operator : $(\underbrace{*\ /\ \%}_{\text{high}}\ \ \underbrace{+\ -}_{\text{Low}})$

1 → * / % ← Same Precedence (Left → Right)

2 → + - ← Same precedence (Left → Right)

Ex $3 * 4 + 6 / 2 - 10 \rightarrow$ *, / ← Same

$12 + \underbrace{6/2} - 10$

$12 + 3 - 10 \rightarrow \underbrace{12+3} - 10$ + - ← Same

$15 - 10 \Rightarrow \boxed{5}$

4) Bitwise Shift operator (<< , >>) (Left to Right)

5) Relational Operator

- 1 → <, <=, >, >= (high) (Left to Right)
- 2 → ==, != (low) (L → R)

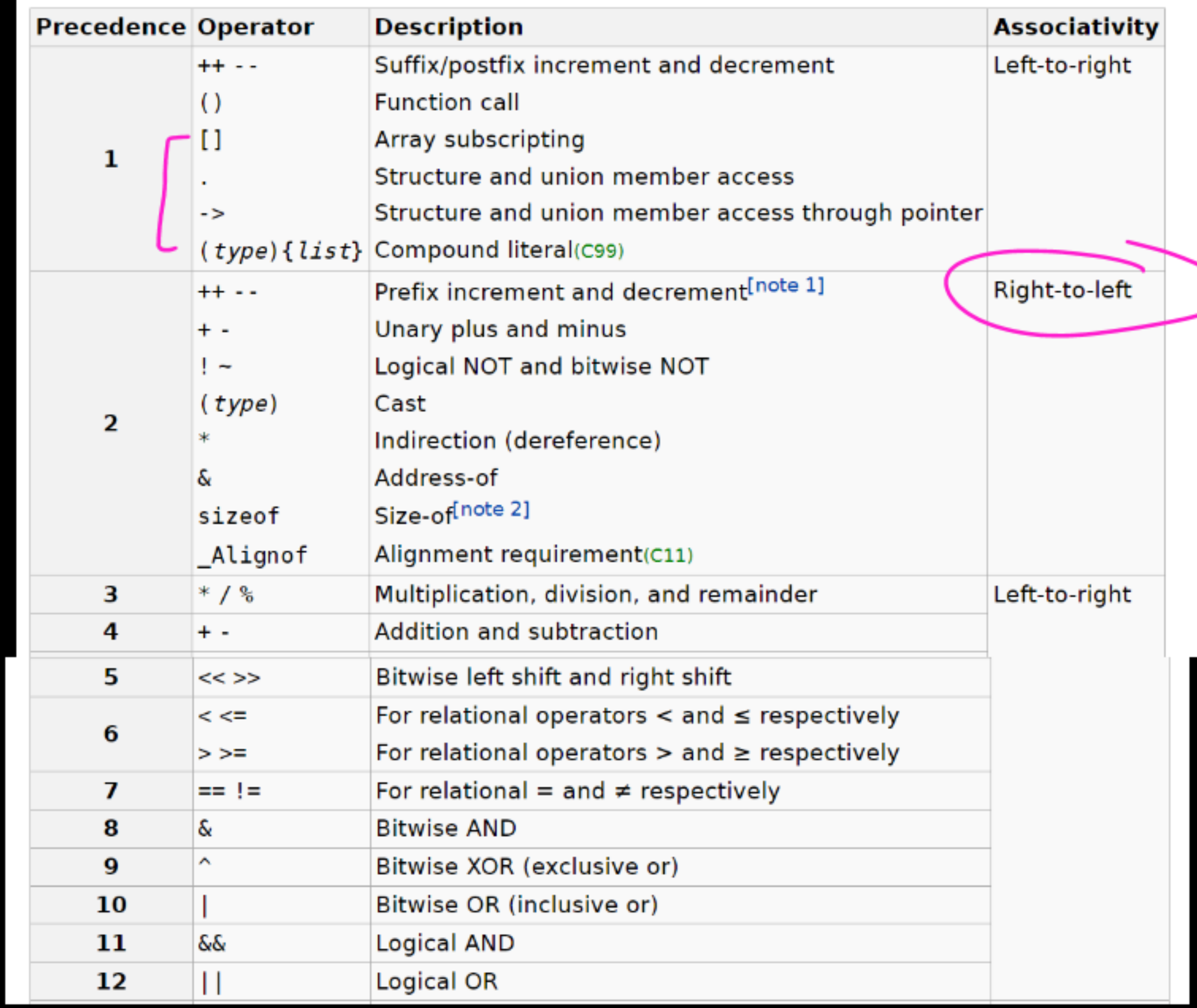
6) Bitwise &

7) Bitwise XOR (^)

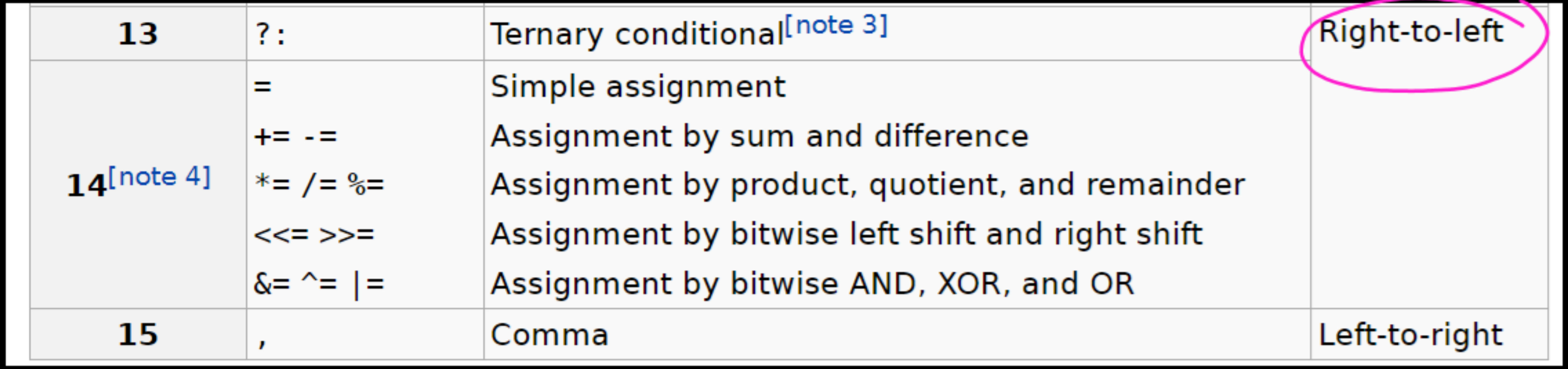
8) Bitwise OR (|)

9) Logical AND (&&)

10) Logical OR (||)



Precedence	Operator	Description	Associativity
1	++ --	Suffix/postfix increment and decrement	Left-to-right
	()	Function call	
	[]	Array subscripting	
	.	Structure and union member access	
	->	Structure and union member access through pointer	
	(*type*){*list*}	Compound literal(C99)	
2	++ --	Prefix increment and decrement[note 1]	Right-to-left
	+ -	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(*type*)	Cast	
	*	Indirection (dereference)	
	&	Address-of	
	sizeof	Size-of[note 2]	
	_Alignof	Alignment requirement(C11)	
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

11) Ternary (Trinary) $\longrightarrow$ Right to left

$$\begin{tikzcd}
\text{12) Assignment (Right to Left)} \arrow[r] \arrow[dr] \arrow[ddr] & = \\
& +=,\ -= \\
& *=,\ /=,\ \%=
\end{tikzcd}$$

13) Comma (Left to Right)

$; \longrightarrow ;$

Between 1 Semicolumn, It is not allowed to Modify a value twice and Can not assign also.