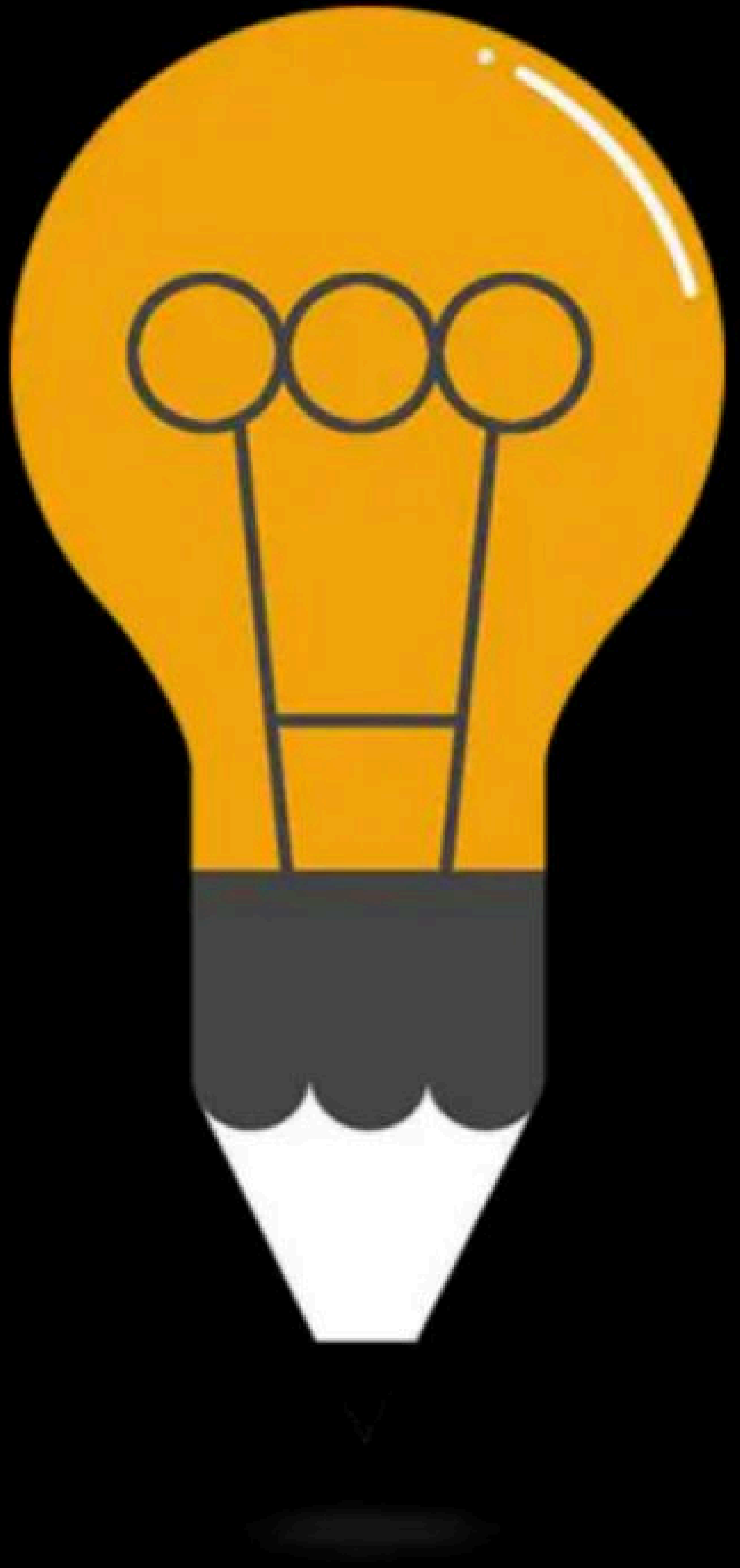
C-Tokens and Statements

Course on C-Programming & Data Structures: GATE - 2024 & 2025

Vishvadeep Gothi • Lesson 2 • Nov 25, 2022



C-Tokens

By: Vishvadeep Gothi

C-tokens

a - z

A - Z

0 - 9

special char.

C-tokens

✓ 1. Keywords

✓ 2. Identifiers

3. Operators

✓ 4. Literals (Constants)

Keywords

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	volatile
const	float	short	unsigned

Identifiers

name of variables, functions, arrays, pointers

Rules :-

1. It should not be a keyword
2. It should contain A to Z a to z 0 to 9 underscore(_)
3. It should not start with a digit

✓ a1 ✓ temp_char ✗ 1apple

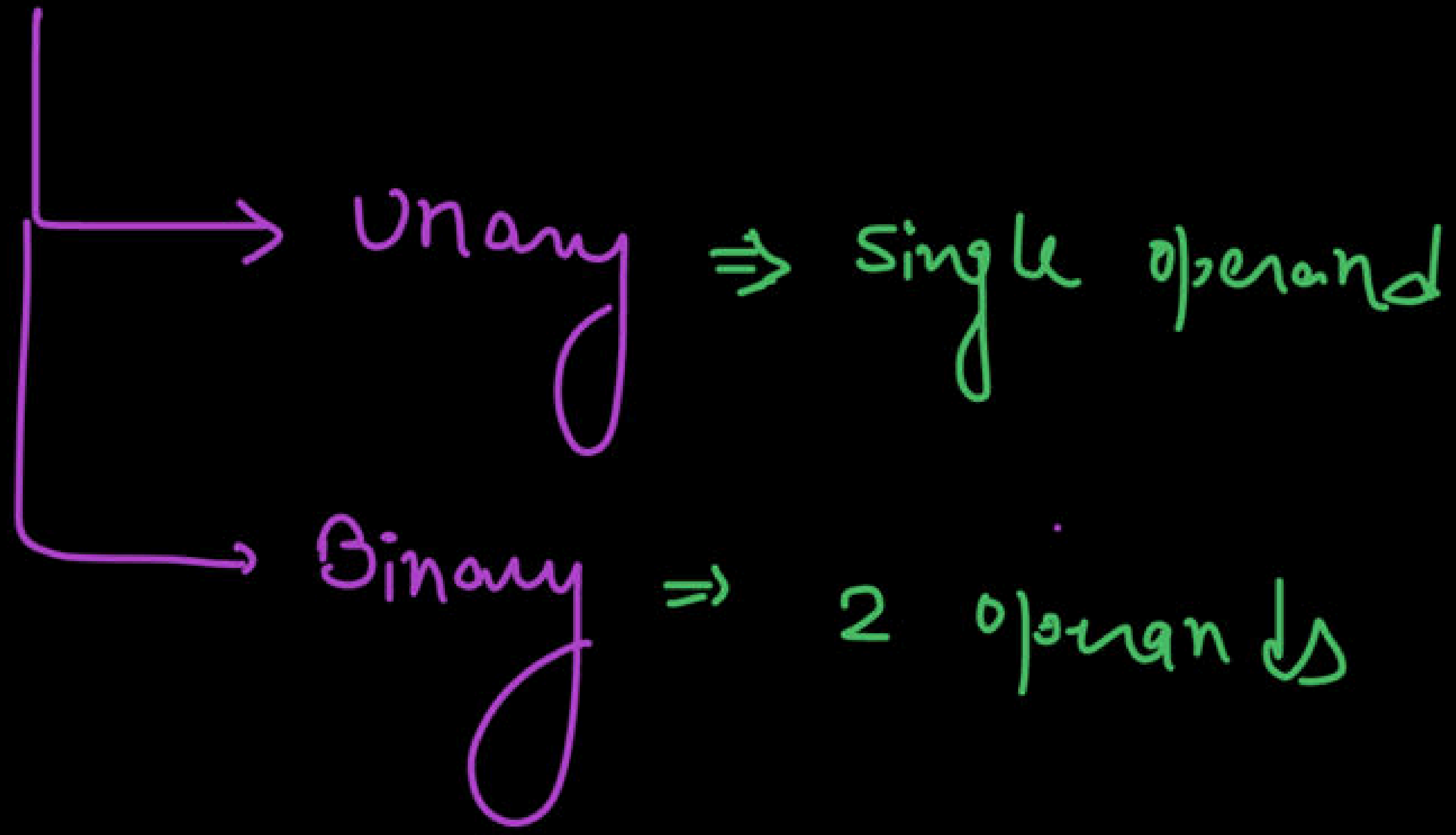
✓ a2 ✓ ch1_fine

Literals

constants, whos value can not be changed during prog. execut^n-

Operators

1. Arithmetic
2. Logical
3. Relational
4. Bitwise Operators
5. Assignment Operators
6. Conditional Operator
7. Termination Operator
8. Special Operators



Unary ⇒ single operand

Binary ⇒ 2 operands

Arithmetic Operators

+	addition
-	subtraction
*	multiplication
/	Division
%	modules ⇒ a % b
++	increment
--	decrement

divide a by b and returns remainder

2+3 ⇒ 5
3 - 1 ⇒ 2
3 * 5 ⇒ 15
6/2 ⇒ 3

```
3) 11 (3
    9
   ---
    2  ← Remainder
```

11 % 3 ⇒ 2
15 % 5 ⇒ 0
20 % 4 ⇒ 0
19 % 7 ⇒ 5
3 % 5 ⇒ 3

-5 % 3 = 1

-13 % 4 = 3

$a / b \Rightarrow Q\frac{r}{b}$

$Q * b + r = a$

$19 \% 7 \Rightarrow 5$

$19/7 \Rightarrow 2$

$\frac{19}{7} \Rightarrow 2\frac{5}{7}$

$\frac{-5}{3}$

$-2 * 3 + \underline{1} = -5$

$Q = -2$

$r = \underline{1}$

$a = 5$

$a{+}{+} \Rightarrow a = 6$

↳ $a = a + \underline{1}$

$a = 4$

$a{-}{-} \Rightarrow a = 3$

Operator Precedence

$\underline{2+3}*5$ ⇒ correct $2+\underline{3*5}$

✗ $\underline{5*5}$ $2+15$

25 17

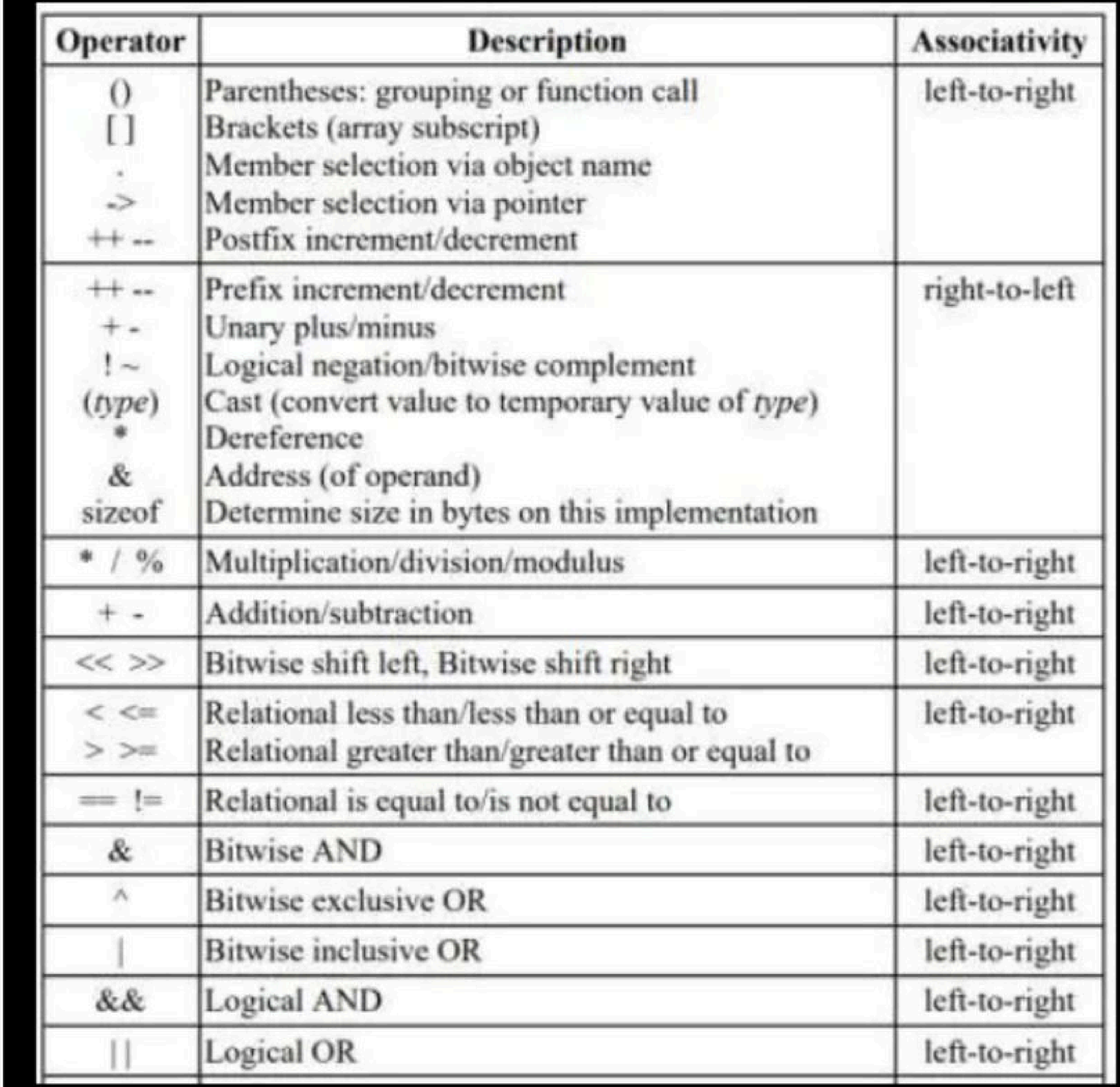
$2-\boxed{3-5}$ $\underline{2-3}-5$ associativity ⇒ left to right

⇒ $2-(-2)$ × $\underline{-1-5}$

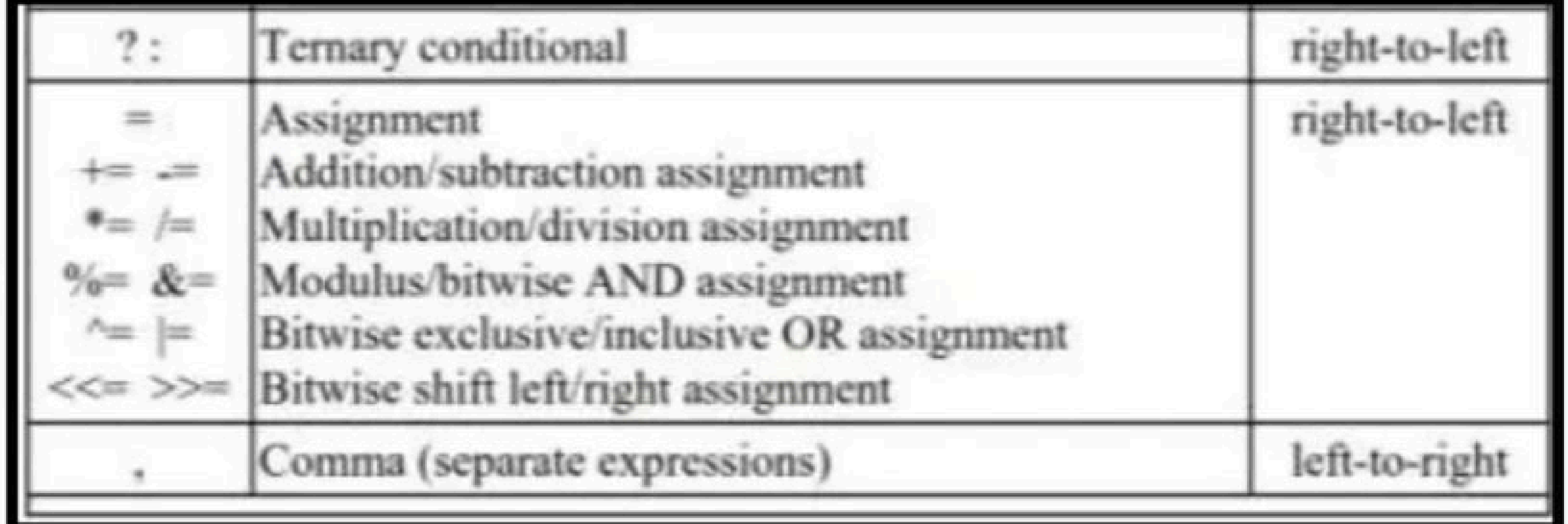
⇒ 4 × -6 Ans

Operator Associativity

Operator Precedence Associativity



Operator	Description	Associativity
() [] . -> ++ --	Parentheses: grouping or function call Brackets (array subscript) Member selection via object name Member selection via pointer Postfix increment/decrement	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

Logical Operators

&&	logical AND
||	logical OR
!	logical NOT

AND			OR	X-OR
A	B	A AND B	A OR B	A X-OR B
F	F	False	F	F
F	T	False	T	T
T	F	False	T	T
T	T	True	T	F

a = 5
b = 2

!(a > b)
⇓
false

a = 5
b = 3
c = 2

(a > b) && (b > c) ⇒ True

(a > b) && (c > a) ⇒ False

(a > b) || (c > a) ⇒ True

!((a > b) || (c > a)) ⇒ False

Relational Operators

<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
==	equal to
!=	not equal to

result ⇓ True (1) or False (0)

a = 8
b = 5

a < b	False
a > b	True
a < = b	False
a > = b	True
a = = b	False
a ! = b	True

Bitwise Operators

works on binary

input to any logical operat^n

zero => false

non-zero => True

&	bitwise AND
|	bitwise OR
^	bitwise X-OR
<<	Left shift
>>	Right shift

a = 2 0010

b = 3 0011

a && b => 1 as true

a || b => 1

a & b => 2

a | b => 3

```
&  0010          OR  0010
   0011              0011
   ----              ----
   0010 => 2         0011
```

$a \wedge b \Rightarrow \underline{1}$

```
^  0010
   0011
  ------
   0001
```

left shift :- <<

$a << b \Rightarrow$ shift left binary of a, b times

$2 << 3 \Rightarrow 16$

$a << b \Rightarrow a * 2^b$

binary of 2 in 8-bits ⇒ $\underline{0}0000010$

```
             00000100

             00001000

16  ⇐       00010000
```

Right shift:-

$a >> b$ ⇒ shift binary of a right side b times

$14 >> 2 \Rightarrow 3$

binary of 14 ⇒ 00001110

00000111

00000011 ⇒ 3

$a >> b$

$$\left\lfloor \frac{a}{2^b} \right\rfloor$$

$\lfloor \quad \rfloor$ floor $\qquad \lfloor 3.1 \rfloor \Rightarrow 3 \quad \lfloor 3.9 \rfloor \Rightarrow 3$

$\lceil \quad \rceil$ ceil $\qquad \lceil 3.1 \rceil \Rightarrow 4 \quad \lceil 3.9 \rceil \Rightarrow 4$

Assignment Operators

$a = \cancel{5}3$
$b = 3$

= assignment

+= $a += b \Rightarrow a = a + b$

-=

*= $a /= b$ $a = a/b$

/=

&=

%=

⋮

$a = b$

$a =$ expression

$\Downarrow$

$a = 2 + 3 * 5$

expression $= a \Leftarrow$ error

$b + c = a$

Conditional Operator

Special Operators

*

&

->

.

sizeof

(type)

Data Types

Types	Data Types
Basic Data Type	int, char, float, double
Derived Data Type	array, pointer, structure, union
Enumeration Data Type	enum
Void Data Type	void

Data Types

Data Types	Modifiers
int	short
char	long
float	unsigned
double	signed
void	

Variables

Output Function

Input Function

Header File

Structure of Program in C-language

Happy Learning.!