Arithmetic Operators

+ − * / %

→ Binary operators

% → To compute remainder.

Exponentiation operation

Library function : pow(2,5); <math.h>

Operand 1	Operator	operand 2
?	+	? → int, float, char, double
?	−	? → " " " "
?	*	? → "
?	/	? → "
?	%	? → It should not be zero ? → It should not be a float/double type.

float op double → double

float op int → float

int op long int → long int

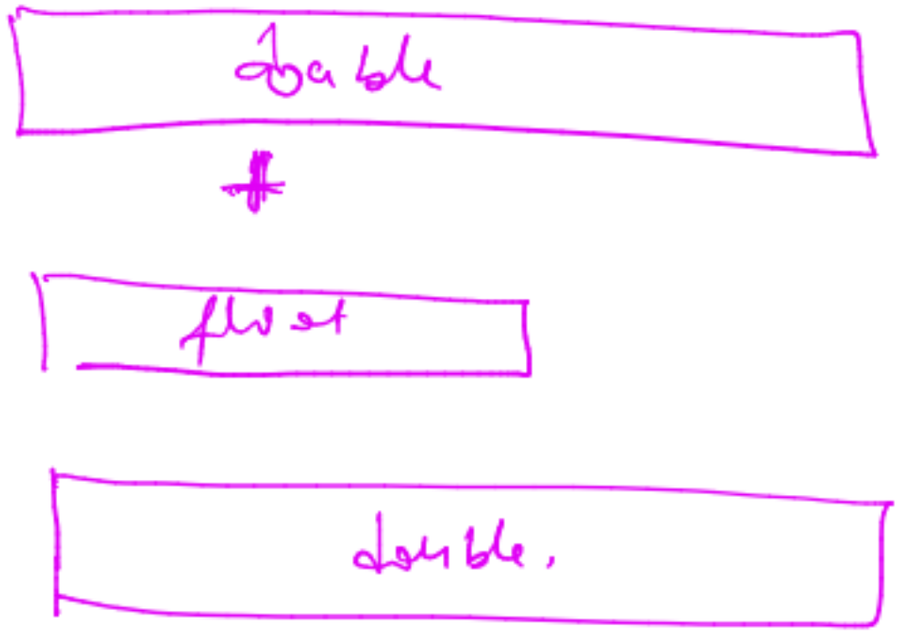
int 2 bytes.

-32768 to +32767

→ 32765 + 3 = -32768

int int int

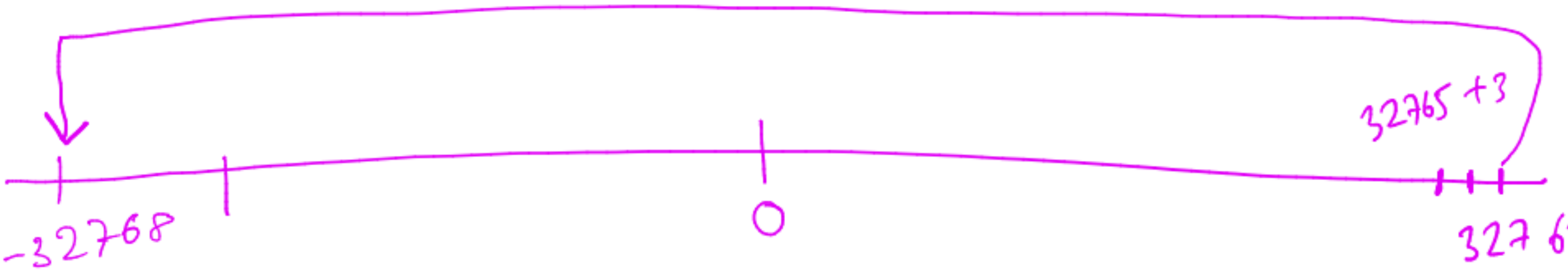
int 4 bytes



98765 4

98765 1 + 4

32765 + 3 = -32768



-32766 -3

char c, a, b;

a = 'A'; ⟶ 65

b = 'B'; ⟶ 66

c = a+b;

c = ?

131

signed ⟶ 1 byte

-2^7 to 2^7-1

-128 to 127

62

a = ' 1 ' ; ⟶ 10

b = ' 2 ' ; ⟶ 12

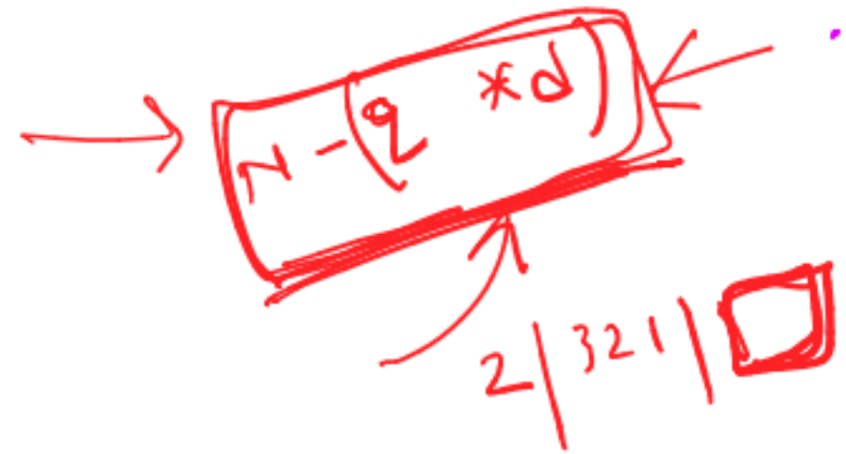
c = a+b; ⟶ 22

%c ⟶ character.

%d ⟶ integer/ASCII

0 to 127

```
int a = 10 ;
float b = 4.0 ;
int c ;
```




```
c = [(a+b)] %5 ;
```

→ Error

↑ ↑
int float.

we cannot have a float as one of the operand.

→ float

```
c = (int) (a+b) %5 ;
```

← fine

→ casting operator

10 f 4.0 : 14 | 0

↙ cast

14

```
c = a %5 ;
  = (int) a %5 ;
```

```
(int) a % (int) b ;
```

```
(int) (a %5) ;
```

→ (char) (a %. 5) ;

[* / %] > [+ -]

Associativity
L → R.

↑ precedence ↓

e = a - b / c * d ;

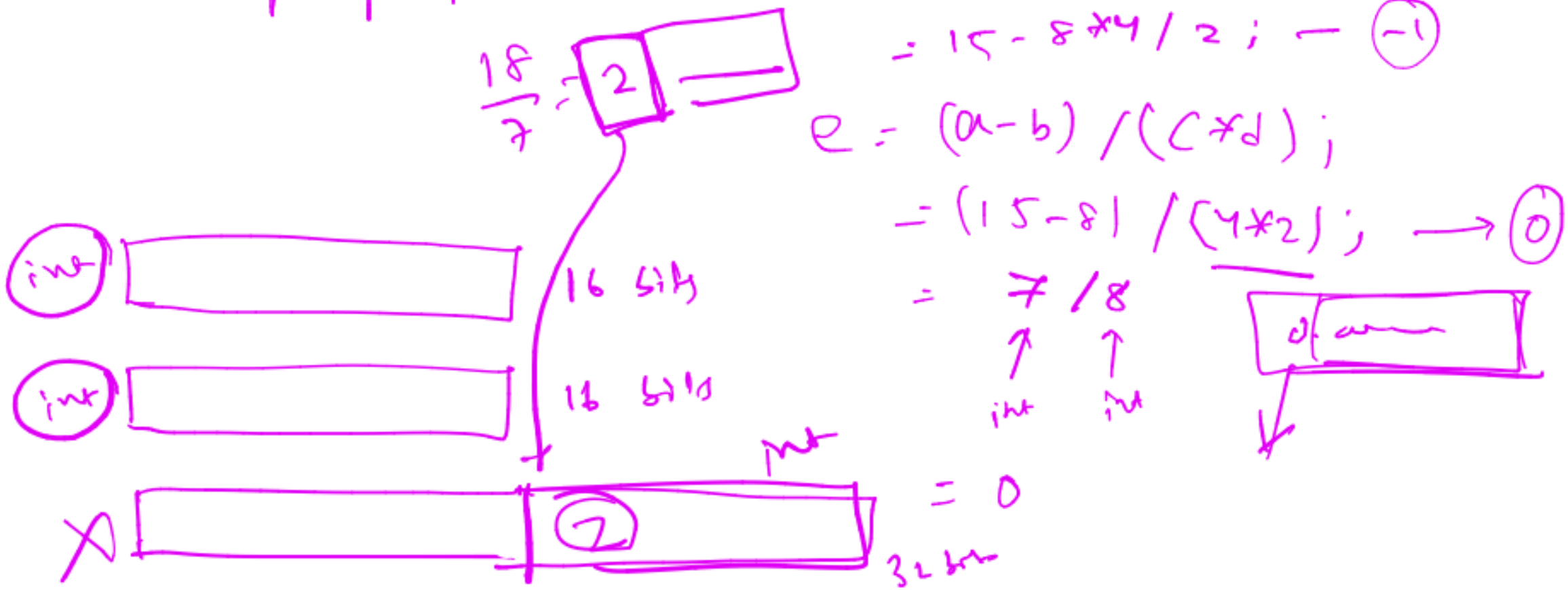
int a=15, b=8, c=4, d=2

e = a - b / c * d ;
= 15 - 8/4 * 2 = (11)

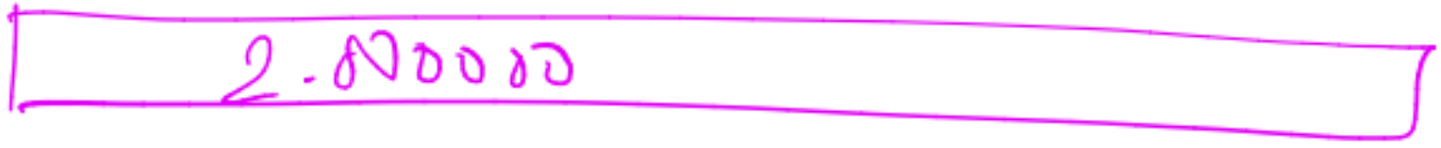
e = a - b * c / d ;

e = a - b * c / d ;
= 15 - 8 * 4 / 2 ; → (-1)

e = (a-b) / (c*d) ;
= (15-8) / (4*2) ; → (0)
= 7 / 8
(int int)
= 0



float e



32 bits

int a → 18

int b → 7

e = a / b;

= 18 / 7;

int int

→ (2)