20/7/23

* OPERATORS:

Symbols that tell compiler to perform some operation

(1.) → <u>Arithmetic Operators:</u>

<u>Binary</u>	<u>Unary</u>
+	++
-	--
*	
/	
%	

(2.) → <u>Relation Operators:</u>
↳ Boolean value.

<u>Operators:</u>

==

!=

>

<

>=

<=

(3.) → <u>Logical Operators</u>

<u>Operators:</u>

&& (Logical AND)

|| (Logical OR)

! (Logical NOT)

* **Binary Number System: (Base 2)**

- **4 to Binary:**

  $$\downarrow$$

  $$\boxed{100}$$

2	4	0
2	2	0
	1	

  remainder.

- **Binary to Decimal:**

1	0	0
$\downarrow$	$\downarrow$	$\downarrow$

  $$2^2 + 2^1 + 2^0 \quad -$$
  $$4 + 0 + 0 \qquad = \boxed{4}$$

- **1 0 1 to Decimal.**

1	0	1
$\downarrow$	$\downarrow$	$\downarrow$

  $$2^2 + 2^1 + 2^0$$
  $$4 + 0 + 1 \qquad = \boxed{5}.$$

- **Decimal to binary:**

  $$(101)_2 = (5)_{10}$$

2	5	1
2	2	0
	1	

  $$= \boxed{101}$$

0 → 000	8 → 1000	16 → 10000
1 → 001	9 → 1001	17 → 10001
2 → 010	10 → 1010	
3 → 011	11 → 1011	
4 → 100	12 → 1100	
5 → 101	13 → 1101	
6 → 110	14 → 1110	
7 → 111	15 → 1111	

* **Other Systems:**

Octal : Base 8

Hexadecimal: Base 16.

* **Bitwise Operator:** True → 1, False → 0

Operators: $A = 0101$, $B = 0110$

(1.) **& (Binary AND):**

eg:
```
    0 1 0 1
&   0 1 1 0
─────────────
    0 1 0 0
```

(2.) **| (Binary OR):**

eg:
```
    0 1 0 1
|   0 1 1 0
─────────────
    0 1 1 1
```

(3.) **^ (Binary XOR):**

Similar value ⟶ False ⇒ { 0 0 → 0
                          1 1 → 0

Different value ⟶ True ⇒ { 0 1
                          1 0 } → 1

eg:
```
    0 1 0 1
^   0 1 1 0
─────────────
    0 0 1 1
```

(4.) ~ (Binary One's Complement) :

$$1 \longrightarrow 0 \quad , \quad 0 \longrightarrow 1$$

A = 0 1 0 1
~A = 1 0 1 0

(5.) << (Binary left shift):

Syntax : Number Operator position.

eg: (a) A << 1

0 1 0 1 << 1

0 1 0 1

∴ 1 0 1 0 _ _ ⟹ 1010

(b) B : 0 1 1 0 << 1 ⟹ 1 1 0 0

(6.) >> (Binary Right shift):

eg: (a) A >> 1 ⟹ 0 1 0 1 >> 1 ⟹ 0 0 1 0

(b) B >> 1 ⟹ 0 1 1 0 >> 1 ⟹ 0 0 1 1

* <u>Assignment Operators</u>:

<u>Operator</u>:

=

+ =

− =

* =

/ =