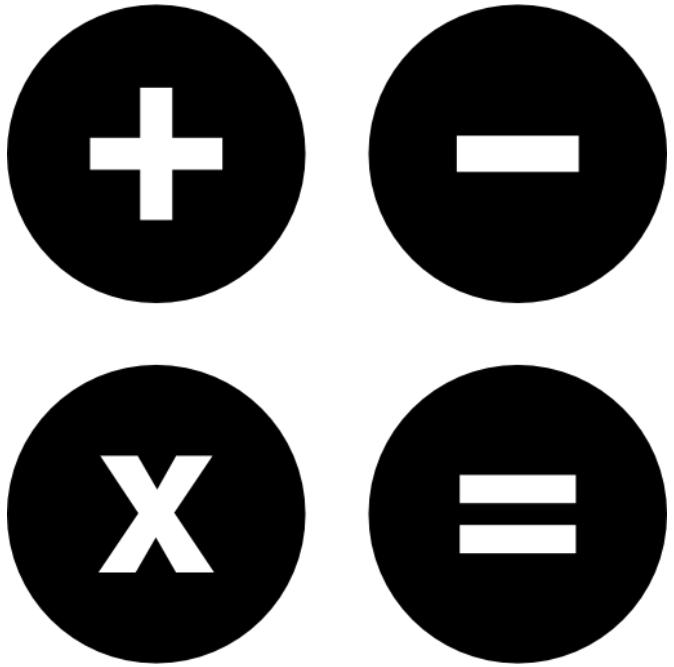
Operator

Jinwoo Choi

Operator(연산자)란?

Data를 계산,조작 하는 기호 또는 키워드

- **Arithmetic Operators** (산술 연산자)
- **Assignment Operators** (대입 연산자)
- **Increment and Decrement Operators** (증감 연산자)
- **Comparison Operators** (비교 연산자)
- **Logical Operators** (논리 연산자)
- **Bitwise Operators** (비트 연산자)
- **Ternary Operator** (삼항 연산자)



Arithmetic Operators(산술 연산자)

Operator	Description	Example	Result (a = 10, b = 3)
+	Addition	a + b	13
-	Subtraction	a – b	7
*	Multiplication	a * b	30
/	Division	a / b	3
%	Modulus	a % b	1

Assignment Operators(대입 연산자)

Operator	Description	Example (a = 10)	New value of a
=	Assign value	a = 5;	5
+=	Add and assign	a += 3;	13
-=	Subtract and assign	a -= 2;	8
*=	Multiply and assign	a *= 2;	20
/=	Divide and assign	a /= 2;	5
%=	Modulus and assign	a %= 3;	1

Increment and Decrement Operators (증감 연산자)

Operator	Description	Example (a = 5)	New value of a, b
++a	Pre-increment	int b = ++a;	a = 6, b = 6
a++	Post-increment	int b = a++;	a = 6, b = 5
--a	Pre-decrement	int b = --a;	a = 4, b = 4
a--	Post-decrement	int b = a--;	a = 4, b = 5

Comparison Operators(비교 연산자)

Operator	Description	Example (a = 10, b = 5)	Result (1 = true, 0 = false)
==	Equal to	a == b	0
!=	Not equal to	a != b	1
>	Greater than	a > b	1
<	Less than	a < b	0
>=	Greater than or equal to	a >= b	1
<=	Less than or equal to	a <= b	0

Logical Operators(논리 연산자)

Operator	Description	Example (x = 1, y = 0)	Result (1 = true, 0 = false)
&&	Logical AND(true if both are true)	x && y	0
||	Logical OR(true if at least one is true)	x || y	1
!	Logical NOT(negates a value)	!x	0

Bitwise Operators(비트 연산자)

Operator	Description	Example (a = 5, b = 3)	Result
&	Bitwise AND	a & b	1 (101 & 011 -> 001)
|	Bitwise OR	a | b	7 (101 | 011 -> 111)
^	Bitwise XOR	a ^ b	6 (101 ^ 011 -> 110)
~	Bitwise NOT	~a	-6 (~00000101 -> 11111010)
<<	Left shift	a << 1	10 (101 << 1 -> 1010)
>>	Right shift	a >> 1	2 (101 >> 1 -> 10)

Ternary Operators(삼항 연산자)

Syntax	Description
(condition) ? true_result : false_result;	A shorthand ‘if-else’ statement

```c
#include<stdio.h>

int main(void)
{

    int a = 10, b = 20;
    int max;

    if(a>b)
    {
        max = a;
    }
    else
    {
        max = b;
    }

    printf("%d\n", max);

    return 0;
}
```

```c
#include<stdio.h>

int main(void)
{

    int a = 10, b = 20;
    int max = (a > b) ? a : b;
    printf("%d\n", max);

    return 0;
}
```

C Operator precedence

```
int a = 10;
int b;
b = a = 5;
int x =  20 / 2 * 5;

int y = 10;
int z = (y+=5) * 2;
```

Rank	Operator in C	Description	Result	Associativity
A	()	Grouping	exp	N/A
B1	()	Function call	rexp	L-R
B2	[]	Subscript	lexp	L-R
B3	.	Structure member	lexp	L-R
B4	->	Structure pointer member	lexp	L-R
B5	++	Postfix increment	rexp	L-R
B6	--	Postfix decrement	rexp	L-R
C1	!	Logical negate	rexp	R-L
C2	~	One's complement	rexp	R-L
C3	+	Unary plus	rexp	R-L
C4	-	Unary minus	rexp	R-L
C5	++	Prefix increment	rexp	R-L
C6	--	Prefix decrement	rexp	R-L
C7	*	Indirection (dereference)	lexp	R-L
C8	&	Address of	rexp	R-L
C9	sizeof	Size in bytes	rexp	R-L
D	(type)	Type conversion (cast)	rexp	R-L
E1	*	Multiplication	rexp	L-R
E2	/	Division	rexp	L-R
E3	%	Integer remainder (modulo)	rexp	L-R
F1	+	Addition	rexp	L-R
F2	-	Subtraction	rexp	L-R
G1	<<	Left shift	rexp	L-R
G2	>>	Right shift	rexp	L-R
H1	>	Greater than	rexp	L-R
H2	>=	Greater than or equal	rexp	L-R
H3	<	Less than	rexp	L-R
H4	<=	Less than or equal	rexp	L-R
I1	==	Equal to	rexp	L-R
I2	!=	Not equal to	rexp	L-R
J	&	Bitwise AND	rexp	L-R
K	^	Bitwise exclusive OR	rexp	L-R
L	|	Bitwise inclusive OR	rexp	L-R
M	&&	Logical AND	rexp	L-R
N	||	Logical OR	rexp	L-R
O	?:	Conditional	rexp	N/A
P1	=	Assignment	rexp	R-L
P2	+=	Add to	rexp	R-L
P3	-=	Subtract from	rexp	R-L
P4	*=	Multiply by	rexp	R-L
P5	/=	Divide by	rexp	R-L
P6	%=	Modulo by	rexp	R-L
P7	<<=	Shift left by	rexp	R-L
P8	>>=	Shift right by	rexp	R-L
P9	&=	AND with	rexp	R-L
P10	^=	Exclusive OR with	rexp	R-L
P11	|=	Inclusive OR with	rexp	R-L
Q	,	Comma	rexp	L-R

LAB – Operators

- Create 'Operators_YourName.c' file.
- Declare 5 variables(can be more than 5) and do at least 10 operations.
- Print out all operation results.
- Think about how to show 1024 with Bitwise Operator for the result.
  - ex – a=5, b=3 -> result = a | b -> result = 7 (101 | 011 -> 111)
- Your code should have
  - Arithmetic Operators
  - Assignment Operators
  - Increment and Decrement Operators
  - Comparison Operators
  - Logical Operators
  - Bitwise Operators
  - Ternary Operator(Optional)
- Before executing the program, write down your expected results and compare them afterward.