제3장
연산자(Operator)

1. 연산자(Operator)는 무엇일까?

- **연산자(Operator)**
  - 어떠한 기능을 수행하는 기호(+,-,*,/ 등)
- **피연산자(Operand)**
  - 연산자의 작업 대상(변수,상수,리터럴,수식)

a + b

+는 이항연산자 → **피연산자가 2개이므로 이항이라고 한다.**

2. 연산자의 종류는?

1) 단항 연산자(부호) : + - ++ -- ~ !
   - 피연산자가 1개임.

2) 이항 연산자
   ① 산술 연산자 : + - * / % << >>
   ② 비교 연산자 : > < >= <= == !=
   ③ 논리 연산자 : && ||
   ④ 비트 연산자 : & | ^

3) 삼항 연산자 : (식) ? 참 : 거짓

4) 대입 연산자 : =
   - 대입연산자만 유일하게 오른쪽에서 왼쪽으로 실행됨

- 연산자 우선 순위 : 단항 -> 이항 -> 삼항 -> 대입
- 이항 연산자 우선 순위 : 산술 -> 비교 -> 논리

3. 연산자의 우선순위는?

1) 괄호가 우선순위가 가장 높다.

2) 산술 > 비교 > 논리 > 대입

3) 단항 > 이항 > 삼항

▶ 명심할 것은, 좌측 우선순위를 힘들여 외울 필요가 없다. 단, 괄호가 제일 우선한다는 것은 이미 다 알고 있다. 프로그래밍에서도 마찬가지다. 가독성을 좋게 하기 위하여 괄호를 쓰는 것이 권장이다.

4) 모든 연산자의 진행방향은 왼쪽 → 오른쪽
대입 연산자만 유일하게 오른쪽 → 왼쪽

1 * 2 * 3

x = y = 10

4. 증감(증가,감소) 연산자란?

1) 증가 연산자(++) : 피연산자의 값을 1 증가 시킨다.

2) 감소 연산자(--) : 피연산자의 값을 1 감소 시킨다.

ex) int i = 10;
    int j = 5;

전위형	j = ++i;	++i; j = i;	값이 참조되기 전에 증가시킨다.
후위형	j = i++;	j = i; i++;	값이 참조된 후에 증가시킨다.

▶세미콜론(;)을 기준으로 하여 전위와 후위의 값이 달라진다는 것에 주목하면 된다. 아울러, 증감연산자는 향후 앞으로 많이 사용되니깐 반드시 알고 있어야 한다.

5. 논리 부정 연산자(!)란?

● 논리 부정 연산자(!) : true를 false로, false를 true로 값이 변환된다.

단, 피연산자가 boolean일 때만, 사용 가능하다.

```
ex) boolean power =   false;
    power = !power;
    power = !power;
```

6. 비트전환 연산자(~)란?

● 정수(10진수)를 2진수로 나타낼 때, 1을 0으로 0은 1로 바꾼다.
(1의 보수 개념)

2진수(binary)	10진수(decimal)
0 0 0 0 1 0 1 0	10
1 1 1 1 0 1 0 1	-11
1 1 1 1 0 1 0 1 0 0 0 0 0 0 0 1	-11 +) 1
1 1 1 1 0 1 1 0	-10 2의 보수

음수를 2진수로 표현하는 방법

7. 이항 연산자의 특징

- 이항 연산자는 연산을 실행하기 전에 피연산자의 타입을 일치시키는 작업을 한다.
  - 디폴트 타입인 int보다 바이트 수가 작은 타입은 int로 변환시킨다. (byte, char, short → int)
  - 정수가 실수와 연산을 하게 되면, 표현 범위가 넓은 타입으로 형변환 된다.
    - ◈ byte + short → int + int → int
    - ◈ char + int → int + int → int
    - ◈ int + float → float + float → float
    - ◈ double + float → double + double → double

7. 이항 연산자의 특징

```
byte b1 = 10;
byte b2 = 20;
byte result = b1 + b2;
```

//Error 이유는?(바이트 크기)

byte + byte → int + int → int

```
byte c = (byte)b1 + b2;
```

// Error

*연산자의 우선 순위 : 단항 > 이항 byte + int = int

```
byte c = (byte)(b1 + b2);
```

* 명시적 형변환, 강제 형변환, Unboxing(객체 단)

8. 나머지 연산자(%), 대입 연산자(=)

1) 나머지 연산자

- 나머지를 반환함.
- 홀수, 짝수 등 배수 검사에 주로 사용함

```
int result = 10 % 3;
```

몫은 3, 나머지는 1

2) 대입 연산자

- 모든 연산자 중, 유일하게 오른쪽에서 왼쪽으로 실행된다.
  단, 왼쪽 피연산자는 상수가 아니어야 한다.

```
final int MAX = 3;
MAX = 10;  // 에러
```

9. 비트 연산자(&, |, ^)

피연산자를 비트단위로 연산한다.

단, 실수형(float, double)은 제외된다.

- OR(|)연산자 : 피연산자 중 어느 한쪽이 1이면 1이다.
- AND(&)연산자 : 피연산자 중 둘 다 1이면 1이다.
- XOR(^)연산자(exclusive or) : 피연산자가 서로 다를 때, 1이다.
  - 배타적 논리합이라고도 한다.

x	y	x | y	x & y	x ^ y
1	1	1	1	0
1	0	1	0	1
0	1	1	0	1
0	0	0	0	0

9. 비트 연산자(&, |, ^)

식	2진수		10진수
3 | 5 = 7		0 0 0 0 0 0 1 1	3
	|)	0 0 0 0 0 1 0 1	5
		0 0 0 0 0 1 1 1	7
3 & 5 = 1		0 0 0 0 0 0 1 1	3
	&)	0 0 0 0 0 1 0 1	5
		0 0 0 0 0 0 0 1	1
3 ^ 5 = 6		0 0 0 0 0 0 1 1	3
	^)	0 0 0 0 0 1 0 1	5
		0 0 0 0 0 1 1 0	6

10. 쉬프트 연산자(<< , >>)

2^n 으로 곱하거나, 나눈 결과를 반환한다.

주로, 속도가 빨라서 그래픽 분야에서 많이 사용한다.

x << n 은 x * 2^n과 같다.

x >> n 은 x / 2^n과 같다.

8 << 2 는 8 * 2^2과 같다.

8 >> 2 는 8 / 2^2과 같다.

8 << 2 의 결과

10. 쉬프트 연산자(<< , >>)

2^n 으로 곱하거나, 나눈 결과를 반환한다.

주로, 속도가 빨라서 그래픽 분야에서 많이 사용한다.

x << n 은 x * 2^n과 같다.

x >> n 은 x / 2^n과 같다.

8 << 2 는 8 * 2^2과 같다.

8 >> 2 는 8 / 2^2과 같다.

8 << 2 의 결과

11. 비교 연산자(< , > , >= , <= , == , !=)

피연산자를 같은 타입으로 변환한 후, 비교를 실행한다.

- 결과 값은 true 또는 false가 된다.

'A' < 'B' → 65 < 66 → true

int * int → int

10.0d == 10.0f → 10.0d == 10.0d → true

기본형(boolean제외)과 참조형에 사용되지만, **참조형에는 ==과 !=만 사용할 수 있다.**

- 참조형은 기본적으로 **주소 비교**가 이루어진다.

* 아무리 성능이 좋은 CPU라고 해도 오차가 필히 존재함

12. 논리 연산자(&&, ||)

피연산자를 반드시 boolean타입이어야 하며, 연산결과 또한 boolean이다.

- &&가 || 보다 우선순위가 높다. 같이 사용되는 경우 괄호를 사용하자

◆ OR연산자(||) : 피연산자 중 어느 한 쪽이 true이면 true이다.

◆ AND연산자(&&) : 피연산자 둘 다 true이면 true이다.

```
i > 3 && i < 5      false
i > 3  ||  i < 0    true
```

```
(x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z')
```

소문자 대문자

* 결과 : x에 저장된 값이 알파벳인지 확인하는 조건

13. 삼항 연산자(조건식 ? :), 복합 대입 연산자

1) 삼항 연산자

조건식의 연산결과가 true이면 '표현식1' 의 결과를 반환하며, false일 경우 '표현식2' 의 결과를 반환한다.(if-else문 대체용)

(조건식) ? 표현식1 : 표현식2

```
int score = 70;

char grade = score >= 90 ? 'A' : (score >= 80? 'B' : 'C');
```

2) 복합 대입 연산자

복합 대입 연산자는 코드를 효율적으로, 줄일 수 있는 연산자이다.

종류 : += , -= , *= , /= 등

sum += I; 좌측 문장과 동일한 코드는 sum = sum + I; 이다.(누적값을 구할 때)

아주 많이 사용되니 반드시 알아 두어야 한다.

감사합니다.