

3. 연산자

1. 연산자
2. 연산자 우선순위
3. 연산자 종류

Contents

❖ 목차

- 1절. 연산자와 연산식
- 2절. 연산의 방향과 우선 순위
- 3절. 단항 연산자
- 4절. 이항 연산자
- 5절. 삼항 연산자(? :)

1절. 연산자와 연산식

❖ 연산이란?

- 데이터를 처리하여 결과를 산출하는 것
- 연산자(Operations)
 - 연산에 사용되는 표시나 기호(+, -, *, /, %, =, ...)
- 피연산자(Operand): 연산 대상이 되는 데이터(리터럴, 변수)
- 연산식(Expressions)
 - 연산자와 피연산자를 이용하여 연산의 과정을 기술한 것

1절. 연산자와 연산식

❖ 연산자의 종류 (p.64)

연산자 종류	연산자	피연산자 수	산출값 타입	기능 설명
산술	+, -, *, /, %	이항	숫자	사칙연산 및 나머지 계산
부호	+, -	단항	숫자	음수와 양수의 부호
문자열	+	이항	문자열	두 문자열을 연결
대입	=, +=, -=, *=, /=, %= &=, ^=, =, <<=, >>=, >>>=	이항	다양	우변의 값을 좌변의 변수에 대입
증감	++, --	단항	숫자	1 만큼 증가/감소
비교	==, !=, >, <, >=, <=, instanceof	이항	boolean	값의 비교
논리	!, &, , &&,	단항 이항	boolean	논리적 NOT, AND, OR 연산
조건	(조건식) ? A : B	삼항	다양	조건식에 따라 A 또는 B 중 하나를 선택
비트	~, &, , ^	단항 이항	숫자 blooean	비트 NOT, AND, OR, XOR 연산
쉬프트	>>, <<, >>>	이항	숫자	비트를 좌측/우측으로 밀어서 이동

2절. 연산의 방향과 우선 순위

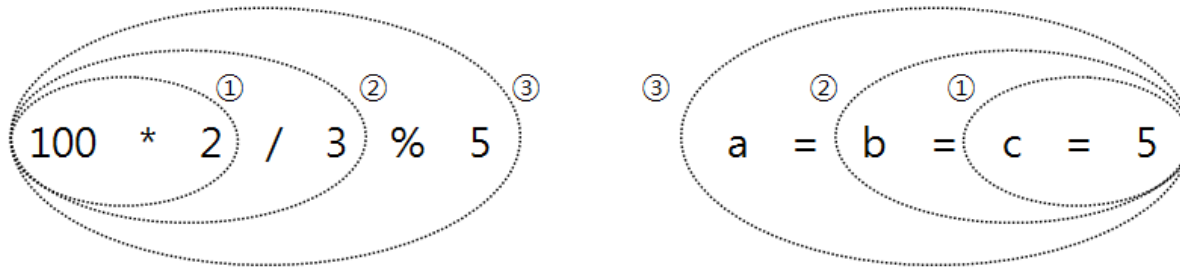
❖ 연산의 방향과 우선 순위

- 연산자의 우선 순위에 따라 연산된다.

```
x > 0 && y < 0
```

- 동일한 우선 순위의 연산자는 연산의 방향 존재

*, /, %는 같은 우선 순위를 갖고 있다. 이들 연산자는 연산 방향이 왼쪽에서 오른쪽으로 수행된다. 100 * 2가 제일 먼저 연산되어 200이 산출되고, 그 다음 200 / 3이 연산되어 66이 산출된다. 그 다음으로 66 % 5가 연산되어 1이 나온다.



하지만 단항 연산자(++ , -- , ~ , !), 부호 연산자(+ , -), 대입 연산자(= , += , -= , ...)는 오른쪽에서 왼쪽(←)으로 연산된다. 예를 들어 다음 연산식을 보자.

2절. 연산의 방향과 우선 순위

❖ 연산의 방향과 우선 순위

연산자	연산 방향	우선 순위
증감(++ , --), 부호(+, -), 비트(~), 논리(!)	←	<div>높음</div> <div>↑</div> <div>↓</div> <div>낮음</div>
산술(*, /, %)	→	
산술(+, -)	→	
쉬프트(<<, >>, >>>)	→	
비교(<, >, <=, >=, instanceof)	→	
비교(==, !=)	→	
논리(&)	→	
논리(^)	→	
논리()	→	
논리(&&)	→	
논리()	→	
조건(?:)	→	
대입(=, +=, -=, *=, /=, %=, &=, ^=, =, <<=, >>=, >>>=)	←	

3절. 단항 연산자

❖ 단항연산자란?

- 피연산자가 1개인 연산자

❖ 단항 연산자의 종류 (p.68~)

- 부호 연산자: +, -
 - boolean 타입과 char 타입을 제외한 기본 타입에 사용 가능
 - 부호 연산자의 산출 타입은 int
- 증감 연산자: ++, --
 - 변수의 값을 1증가 시키거나 (++) 1 감소 (--) 시키는 연산자
 - 증감 연산자가 변수 뒤에 있으면 다른 연산자 먼저 처리 후 증감 연산자 처리

3절. 단항 연산자

❖ 단항 연산자의 종류

- 논리 부정 연산자: !
 - Boolean type 에만 사용가능

연산식		설명
!	피연산자	피연산자가 true 이면 false 값을 산출 피연산자가 false 이면 true 값을 산출

- 비트 반전 연산자: ~
 - byte, short, int, long 타입만 피연산자가 될 수 있다.
 - 비트값을 반전(0 → 1, 1 → 0)시킨다.

연산식		설명
~	10 (0 0 ... 0 1 0 1 0)	산출결과: -11 (1 1 ... 1 0 1 0 1)

4절. 이항 연산자

❖ 이항 연산자란? (p.75~)

- 피연산자가 2개인 연산자

- 종류

- 산술 연산자: +, -, *, /, %
- 문자열 연결 연산자: +
- 대입 연산자: =, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=
- 비교 연산자: <, <=, >, >=, ==, !=
- 논리 연산자: &&, ||, &, |, ^, !
- 비트 논리 연산자: &, |, ^
- 비트 이동 연산자: <<, >>, >>>

4절. 이항 연산자

❖ 산술 연산자

- boolean 타입을 제외한 모든 기본 타입에 사용 가능
- 결과값 산출할 때 Overflow 주의
- 정확한 계산은 정수를 사용
- NaN과 Infinity 연산은 주의할 것

연산식			설명
피연산자	+	피연산자	덧셈 연산
피연산자	-	피연산자	뺄셈 연산
피연산자	*	피연산자	곱셈 연산
피연산자	/	피연산자	좌측 피연산자를 우측 피연산자로 나눴셈 연산
피연산자	%	피연산자	좌측 피연산자를 우측 피연산자로 나눈 나머지를 구하는 연산

```
int result = num % 3;
```

0, 1, 2 중의 한 값

num 을 3 으로 나눈 나머지

4절. 이항 연산자

❖ 문자열 연산자 (p.86)

- 피연산자 중 문자열이 있으면 문자열로 결합

```
String str1 = "JDK" + 6.0;
```

```
String str2 = str1 + " 특징";
```

```
System.out.println(str2);
```



```
String str3 = "JDK" + 3 + 3.0;
```

```
String str4 = 3 + 3.0 + "JDK";
```

```
System.out.println(str3);
```

```
System.out.println(str4);
```

【실행 결과】

 Console 

```
<terminated> Stri  
JDK6.0 특징  
JDK33.0  
6.0JDK
```

4절. 이항 연산자

❖ 비교 연산자(==, !=, <, >, <=, >=) (p.87~91)

- 대소(<, <=, >, >=) 또는 동등(==, !=) 비교해 boolean 타입인 true/false 산출

구분	연산식			설명
동등 비교	피연산자	==	피연산자	두 피 연산자의 값이 같은지를 검사
	피연산자	!=	피연산자	두 피 연산자의 값이 다른지를 검사
크기 비교	피연산자	>	피연산자	피 연산자 1 이 큰지를 검사
	피연산자	>=	피연산자	피 연산자 1 이 크거나 같은지를 검사
	피연산자	<	피연산자	피 연산자 1 이 작은지를 검사
	피연산자	<=	피연산자	피 연산자 1 이 작거나 같은지를 검사

- 동등 비교 연산자는 모든 타입에 사용
- 크기 비교 연산자는 boolean 타입 제외한 모든 기본 타입에 사용
- 흐름 제어문인 조건문(if), 반복문(for, while)에서 주로 이용
 - 실행 흐름을 제어할 때 사용

4절. 이항 연산자

❖ 논리 연산자 (&&, ||, &, |, ^, !) (p.91~93)

- 논리곱(&&), 논리합(||), 배타적 논리합(^), 논리 부정(!) 연산 수행
- 피연산자는 boolean 타입만 사용 가능

구분	연산식			결과	설명
AND (논리곱)	true	&& 또는 &	true	true	피 연산자 모두가 true 일 경우에만 연산 결과는 true
	true		false	false	
	false		true	false	
	false		false	false	
OR (논리합)	true	 또는 	true	true	피 연산자 중 하나만 true 이면 연산 결과는 true
	true		false	true	
	false		true	true	
	false		false	false	
XOR (배타적 논리합)	true	^	true	false	피 연산자가 하나는 true 이고 다른 하나가 false 일 경우에만 연산 결과는 true
	true		false	true	
	false		true	true	
	false		false	false	
NOT (논리부정)		!	true	false	피 연산자의 논리값을 바꿈
			false	true	

4절. 이항 연산자

❖ 비트 연산자(&, |, ^, ~, <<, >>, >>>) (p.94~98)

- 비트(bit) 단위로 연산 하므로 0과 1이 피연산자
 - 0과 1로 표현이 가능한 정수 타입만 비트 연산 가능
 - 실수 타입인 float과 double은 비트 연산 불가
- 종류
 - 비트 논리 연산자(&, |, ^, ~)
 - 비트 이동 연산자(<<, >>, >>>)

4절. 이항 연산자

❖ 비트 논리 연산자(&, |, ^, ~)

- 피 연산자가 boolean타입일 경우 일반 논리 연산자
- 피연산자가 정수 타입일 경우 비트 논리 연산자로 사용
- 비트 연산자는 피연산자를 int타입으로 자동 타입 변환 후 연산 수행

구분	연산식			결과	설명
AND (논리곱)	1	&	1	1	두 비트가 모두가 1 일 경우에만 연산 결과는 1
	1		0	0	
	0		1	0	
	0		0	0	
OR (논리합)	1		1	1	두 비트 중 하나만 1 이면 연산 결과는 1
	1		0	1	
	0		1	1	
	0		0	0	
XOR (배타적 논리합)	1	^	1	0	두 비트 중 하나는 1 이고 다른 하나가 0 일 경우 연산 결과는 1
	1		0	1	
	0		1	1	
	0		0	0	
NOT (논리부정)		~	1	0	보수
			0	1	

4절. 이항 연산자

❖ 비트 이동 연산자(<<, >>, >>>)

- 정수 데이터의 비트를 좌측 또는 우측으로 밀어 이동시키는 연산 수행

구분	연산식			설명
이동 (쉬프트)	a	<<	b	정수 a 의 각 비트를 b 만큼 왼쪽으로 이동 (빈자리는 0 으로 채워진다.)
	a	>>	b	정수 a 의 각 비트를 b 만큼 오른쪽으로 이동 (빈자리는 정수 a 의 최상위 부호 비트(MSB)와 같은 값으로 채워진다.)
	a	>>>	b	정수 a 의 각 비트를 오른쪽으로 이동 (빈자리는 0 으로 채워진다.)

4절. 이항 연산자

❖ 대입 연산자(=, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=)

- 오른쪽 피연산자의 값을 좌측 피연산자인 변수에 저장
- 모든 연산자들 중 가장 낮은 연산 순위 -> 제일 마지막에 수행
- 종류
 - 단순 대입 연산자
 - 복합 대입 연산자
 - 정해진 연산을 수행한 후 결과를 변수에 저장

4절. 이항 연산자

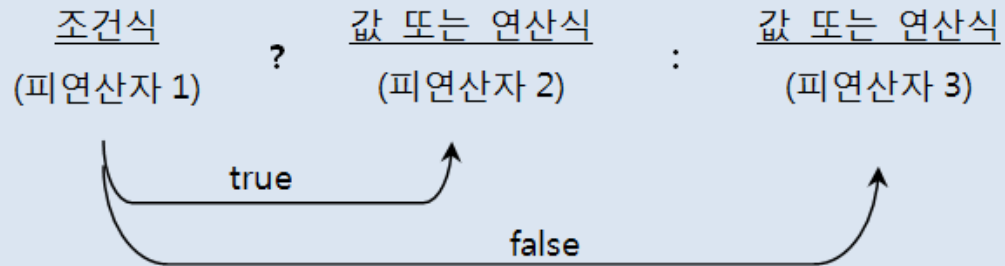
❖ 대입 연산자의 종류

구분	연산식			설명
단순 대입 연산자	변수	=	피연산자	우측의 피연산자의 값을 변수에 저장
복합 대입 연산자	변수	+=	피연산자	우측의 피연산자의 값을 변수의 값과 더한 후에 다시 변수에 저장 (변수=변수+피연산자 와 동일)
	변수	-=	피연산자	우측의 피연산자의 값을 변수의 값에서 뺀 후에 다시 변수에 저장 (변수=변수-피연산자 와 동일)
	변수	*=	피연산자	우측의 피연산자의 값을 변수의 값과 곱한 후에 다시 변수에 저장 (변수=변수*피연산자 와 동일)
	변수	/=	피연산자	우측의 피연산자의 값으로 변수의 값을 나눈 후에 다시 변수에 저장 (변수=변수/피연산자 와 동일)
	변수	%=	피연산자	우측의 피연산자의 값으로 변수의 값을 나눈 후에 나머지를 변수에 저장 (변수=변수%피연산자 와 동일)
	변수	&=	피연산자	우측의 피연산자의 값과 변수의 값을 & 연산 후 결과를 변수에 저장 (변수=변수&피연산자 와 동일)
	변수	=	피연산자	우측의 피연산자의 값과 변수의 값을 연산 후 결과를 변수에 저장 (변수=변수 피연산자 와 동일)
	변수	=	피연산자	우측의 피연산자의 값과 변수의 값을 ^ 연산 후 결과를 변수에 저장 (변수=변수^피연산자 와 동일)
	변수	<<=	피연산자	우측의 피연산자의 값과 변수의 값을 << 연산 후 결과를 변수에 저장 (변수=변수<<피연산자 와 동일)
	변수	>>=	피연산자	우측의 피연산자의 값과 변수의 값을 >> 연산 후 결과를 변수에 저장 (변수=변수>>피연산자 와 동일)
	변수	>>>=	피연산자	우측의 피연산자의 값과 변수의 값을 >>> 연산 후 결과를 변수에 저장 (변수=변수>>>피연산자 와 동일)

5절. 삼항 연산자

❖ 삼항 연산자란?

- 세 개의 피연산자를 필요로 하는 연산자
- 앞의 조건식 결과에 따라 콜론 앞 뒤의 피연산자 선택 -> 조건 연산식



```
int score = 95;  
char grade = (score>90) ? 'A' : 'B'
```

=

```
int score = 95;  
char grade;  
if(score>90) {  
    grade = 'A';  
} else {  
    grade = 'B';  
}
```