연산자와 연산식

- **연산이란?** 주어진 식을 계산하여 결과를 얻어내는 과정
  - 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

연산자와 연산식

❖연산자의 종류

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

연산의 방향과 우선 순위

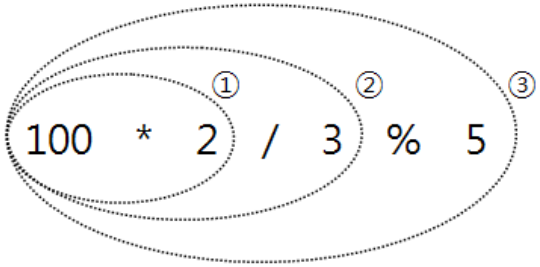
❖연산의 방향과 우선 순위

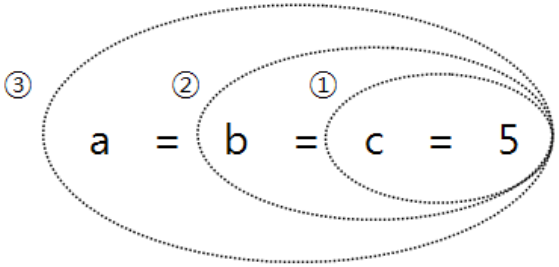
- 연산자의 우선 순위에 따라 연산된다.

```
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 * 2 가 제일 먼저 연산되어 200 이 산출되고, 그 다음 200 / 3 이 연산되어 66 이 산출된다. 그 다음으로 66 % 5 가 연산되어 1 이 나온다.





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

연산의 방향과 우선 순위

❖연산의 방향과 우선 순위

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

단항 연산자

❖단항연산자란?

- 피연산자가 1개인 연산자

❖단항 연산자의 종류

- 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
  - i++
  - ++i

  ] i=i+1

  --i
  i--

  ] i=i-1

단항 연산자

❖단항 연산자의 종류

- 증감 연산자(++,---)

연산자	의미
+x	x를 양수로 만든다.
-x	x를 음수로 만든다.
++x	x값을 먼저 증가한 후에 다른 연산에 사용한다. 이 수식의 값은 증가된 x값이다.
x++	x값을 먼저 사용한 후에, 증가한다. 이 수식의 값은 증가되지 않은 원래의 x값이다.
--x	x값을 먼저 감소한 후에 다른 연산에 사용한다. 이 수식의 값은 감소된 x값이다.
x--	x값을 먼저 사용한 후에, 감소한다. 이 수식의 값은 감소되지 않은 원래의 x값이다.

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.
  - 산출 타입은 int 타입.

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

이항 연산자

- 이항 연산자란?
  - 피연산자가 2개인 연산자
  - 종류
    - 산술 연산자: +, -, *, /, %
    - 문자열 연결 연산자: +
    - 대입 연산자: =, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=
    - 비교 연산자: <, <=, >, >=, ==, !=
    - 논리 연산자: &&, ||, &, |, ^, !
    - 비트 논리 연산자: &, |, ^
    - 비트 이동 연산자: <<, >>, >>>

이항 연산자

❖산술 연산자

- boolean 타입을 제외한 모든 기본 타입에 사용 가능
- 결과값 산출할 때 Overflow 주의
- 정확한 계산은 정수를 사용

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

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

result: 0, 1, 2 중의 한 값

num % 3: num 을 3 으로 나눈 나머지

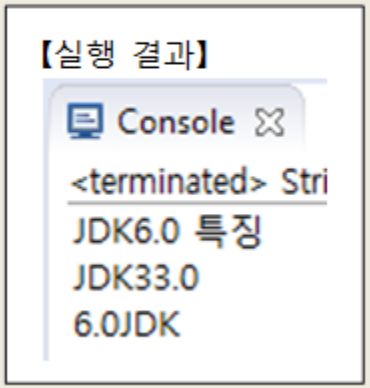
이항 연산자

❖문자열 연산자

- 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】



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

```
int a = 1, b = 3;
a = b;          // b 값을 a에 대입하여 a=3
a += b;         // a = a + b의 연산이 이루어져, a=6. b는 3 그대로
```

이항 연산자

❖대입 연산자의 종류

단순 대입 연산자

대입 연산자	내용	대입 연산자	내용
a = b	b의 값을 a에 대입	a &= b	a = a & b와 동일
a += b	a = a + b와 동일	a ^= b	a = a ^ b와 동일
a -= b	a = a - b와 동일	a |= b	a = a | b와 동일
a *= b	a = a * b와 동일	a <<= b	a = a << b와 동일
a /= b	a = a / b와 동일	a >>= b	a = a >> b와 동일
a %= b	a = a % b와 동일	a >>>= b	a = a >>> b와 동일

복합 대입 연산자

- 대입 연산자(=)는 왼쪽 변수에 오른쪽 수식의 값을 계산하여 저장
- 대입 연산자 == 할당 연산자 == 배정 연산자라고도 한다.

이항 연산자

❖비교 연산자(==, !=, <, >, <=, >=)

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

연산자	내용	예제	결과
a < b	a가 b보다 작으면 true	3<5	true
a > b	a가 b보다 크면 true	3>5	false
a <= b	a가 b보다 작거나 같으면 true	1<=0	false
a >= b	a가 b보다 크거나 같으면 true	10>=10	true
a == b	a가 b와 같으면 true	1==3	false
a != b	a가 b와 같지 않으면 true	1!=3	true

- 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

이항 연산자

❖논리 연산자 (&&, ||, &, |, ^, !)

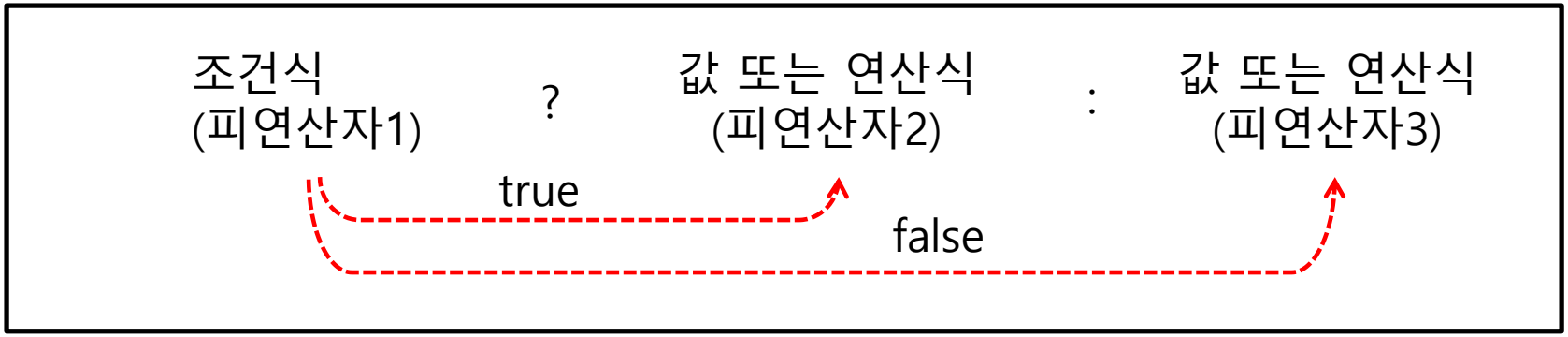
- 논리곱(&&), 논리합(||), 배타적 논리합(^) ,논리 부정(!) 연산 수행
- 피연산자는 boolean 타입만 사용 가능

구분	연산식			결과	설명
AND (논리곱)	true	&& 또는 &	true	true	피 연사자 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	ture	
	false		false	false	
XOR (배타적 논리합)	true	^	true	false	피 연산자가 하나는 ture 이고 다른 하나가 false 일 경우에만 연산 결과는 true
	true		false	true	
	false		true	ture	
	false		false	false	

삼항 연산자

❖삼항 연산자란?

- 세 개의 피연산자를 필요로 하는 연산자.(조건 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';
}
```

형변환

1) 형변환(type cast)은 어떤 자료형의 값을 다른 자료형의 값으로 바꾸어 주는 연산이다.

- 자동적인 형변환
  - 자바는 필요할 때마다 자동적으로 형변환을 한다.
  - double d = 2+3.5;

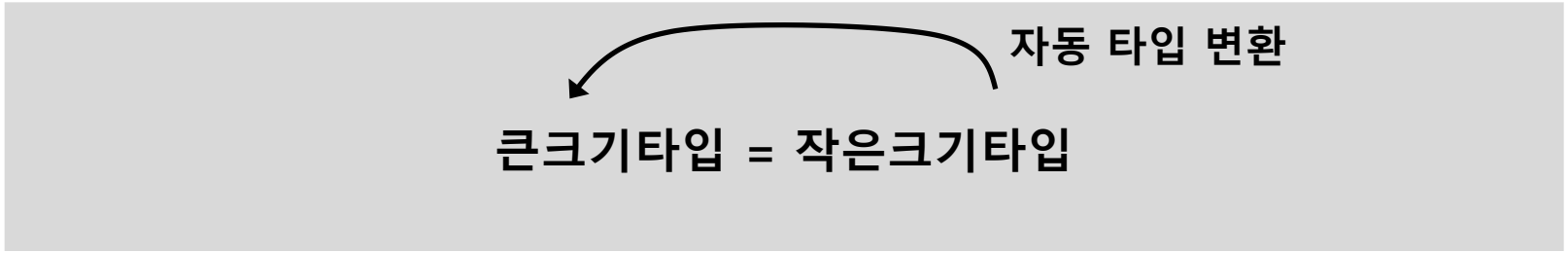
- 축소 변환(narrowing conversion)
  - 더 작은 크기의 자료형에 값을 저장하는 형변환
  - int i = (int) 12.5

- 확대 변환(widening conversion)
  - 확대 변환은 축소 변환의 반대로 더 큰 크기의 변수로 값을 이동하는 변환이다.
  - double d = (double) 100;

형변환

❖자동 타입 변환

- 프로그램 실행 도중 작은 타입은 큰 타입으로 자동 타입 변환 가능



byte(1byte) < short(2) < int(4) < long(8) < float(4) < double(8)

- 연산은 같은 타입의 피연산자(operand)간에만 수행
  - 서로 다른 타입의 피연산자는 같은 타입으로 변환
  - 두 피연산자 중 크기가 큰 타입으로 자동 변환

```
int intValue = 10;
double doubleValue = 5.5;

                double 타입으로 자동 변환
double result = intValue + doubleValue;       //result 에 15.5 가 저장
```

형변환

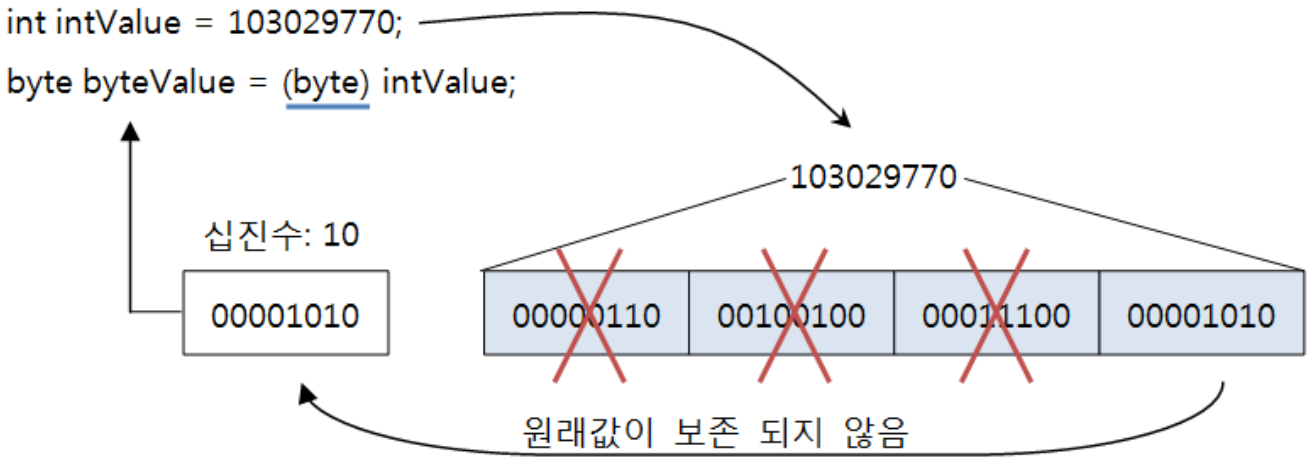
❖강제 타입 변환

- 큰 타입을 작은 타입 단위로 쪼개기
- 끝의 한 부분만 작은 타입으로 강제적 변환

강제 타입 변환

작은크기타입 = (작은크기타입) 큰크기 타입

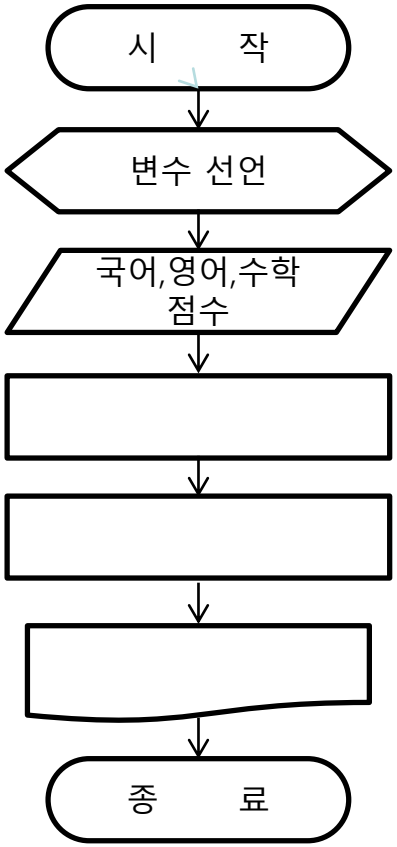
- Ex) int 를 byte에 담기



순차형 문제 1

국어, 영어, 수학 점수를 입력받아, 총점과 평균을 구해 출력하는 프로그램의 순서도와 코드를 작성하시오.

flowchart



pseudocode

```
1    public static void main(String[] args) {




     }
```

실행 예

```
국어 영어 수학 점수를 입력해 주세요 : 90 80 100
총점은 270, 평균은 90.0
```

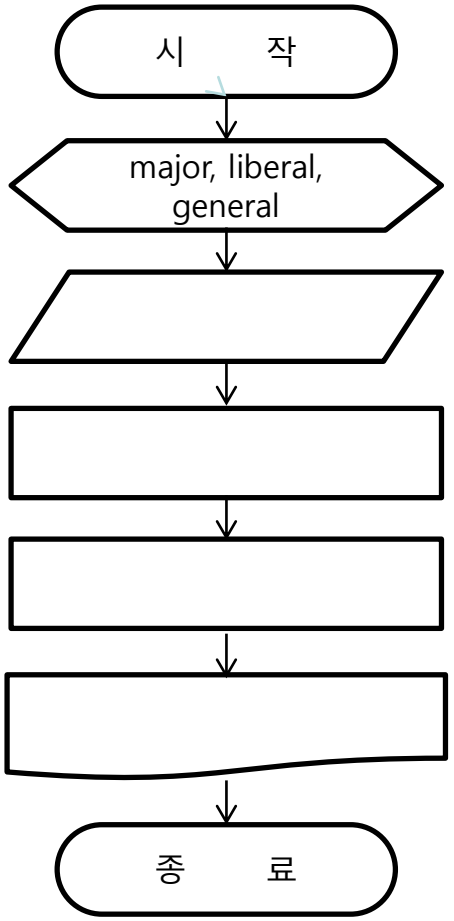
생각해보기

- 총점을 구해야, 평균을 구할 수 있다.
- 평균을 소수점(둘째짜리)까지 구해 출력하도록 수정해보자.

순차형 문제 2

대학을 졸업하려면 최소 140학점을 이수해야 한다고 하자. 이수한 학점 중 전공은 70학점이상이어야 하며, 교양과 일반은 각각 30학점이상이거나 교양과 일반 영역이 80학점 이상이어야한다. 이수한 세개의 학점을 각각 키보드로 입력 받아 졸업여부를 출력하는 순서도와 코드를 작성하시오.

flowchart



pseudocode

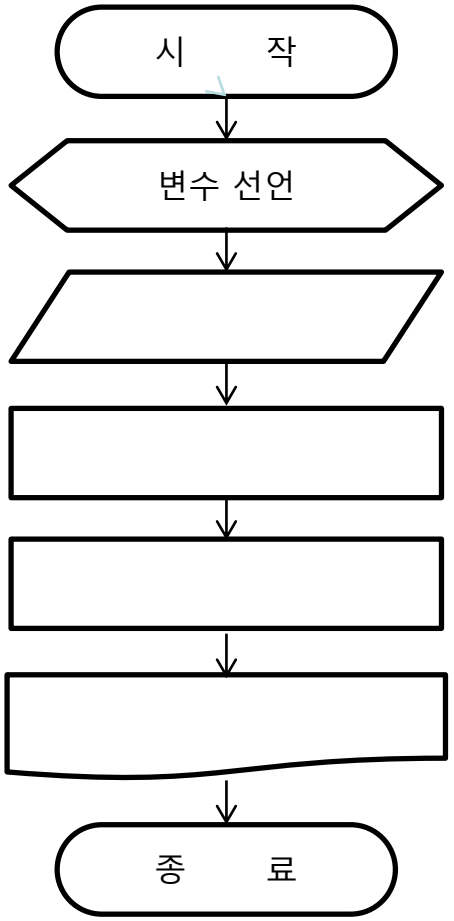
```
public static void main(String[] args) {
        Scanner in = new Scanner(System.in);




}
```

실행 예

순차형 문제 2

대학을 졸업하려면 최소 140학점을 이수해야 한다고 하자. 이수한 학점 중 전공은 70학점이상이어야 하며, 교양과 일반은 각각 30학점이상이거나 교양과 일반 영역이 80학점 이상이어야한다. 이수한 세개의 학점을 각각 키보드로 입력받아 졸업여부를 출력하는 순서도와 코드를 작성하시오.

flowchart



pseudocode

```
1    public static void main(String[] args) {
2        Scanner in = new Scanner(System.in);




}
```

실행 예