컴퓨터 프로그래밍

Operators

Operator, Operand and Expression

- Operator
  - 연산 수행을 위한 약속된 기호
- Operand
  - 연산의 대상이 되는 값
- Expression
  - Operator와 Operand의 조합

25 + 8

Operators

- Operand의 수에 따라
  - unary / binary / ternary
- 연산 종류에 따라
  - Arithmetic operators
  - Relational operators
  - Logical operators
  - Bit operators
  - Assignment operators
  - Incremental / decremental operator
  - conditional operator

- 중요!
  - 모든 연산자는 반환값(return value)이 있다

Arithmetic Operators (산술 연산자)

- 사칙연산 등의 산술 연산
- 연산 결과를 반환

연산	연산자	자료형	결과값
단항 플러스	+a	정수형, 부동소수형	a의 값 그대로
단항 마이너스	-a	정수형, 부동소수형	a의 부호를 변경한 값
덧셈	a + b	정수형, 부동소수형	a와 b의 합
뺄셈	a - b	정수형, 부동소수형	a에서 b를 뺀 값
곱셈	a * b	정수형, 부동소수형	a와 b의 곱
나눗셈	a / b	정수형	a를 b로 나눈 몫
		부동소수형	a를 b로 나눈 값
나머지	a % b	정수형	a를 b로 나눈 나머지

Example

```
class ArithOp
{
    public static void main(String[] args)
    {
        int n1=7;
        int n2=3;

        int result=n1+n2;
        System.out.println("덧셈 결과 : " + result);

        result=n1-n2;
        System.out.println("뺄셈 결과 : " + result);
        System.out.println("곱셈 결과 : " + n1*n2);
        System.out.println("나눗셈 결과 : " + n1/n2);
        System.out.println("나머지 결과 : " + n1%n2);
    }
}
```

```
덧셈 결과 : 10
뺄셈 결과 : 4
곱셈 결과 : 21
나눗셈 결과 : 2
나머지 결과 : 1
```

Operation between Different Data Types

- 서로 다른 data type variables 간의 연산
  - 저장공간의 크기가 작은 변수를 큰 쪽으로 type conversion
- 한 개 혹은 두 개의 피연산자가 부동소수형이면,
  - 연산 결과는 부동소수형
  - e.g.
    - $2 + 3.14 \Longrightarrow 2.0 + 3.14 = 5.14$
    - $12.5 / 5 \Longrightarrow 12.5 / 5.0 = 2.5$

- 정수 / 정수의 결과는 정수(몫), 정수 / 부동소수의 결과는 부동소수

Type conversion

- Implicit
  - Widening conversion
    - 서로 다른 data type variables 간의 연산시 저장공간의 크기가 작은 변수를 큰 쪽으로 type conversion
  - Assignment conversion
    - l-value의 data type과 다른 data type의 r-value를 대입하는 경우 l-value의 data type에 맞추어 type conversion
- Explicit (type casting)
  - 명시적인 형 변환
  - *(data_type) variable / expression*
  - e.g.
```
int a;
float b;
a = (int)b;
```

Precedence

- 연산자 우선순위
  - 한 수식 내에서 연산자의 계산 순서

우선순위	연산자
1순위	단항 + 단항 -
2순위	* / %
3순위	+ -

  - 괄호를 이용한 연산 순서 조정
    - (a - b) * 5.0 / 9.0

      vs.

      a - b * 5.0 / 9.0

Associativity

- 연산자 결합순서
  - 같은 우선순위의 두 연산자가 나란히 있을 때 연산 순서

- 단항 연산자
  - 우측에서 좌측으로 (right associativity)
  - e.g.
    - - + - a $\Longrightarrow$ - (+ (- a))
- 이항 연산자
  - 좌측에서 우측으로 (left associativity)
  - e.g.
    - a / b * c $\Longrightarrow$ (a / b) * c

Relational Operators (관계 연산자)

- 두 값의 동치 / 대소 관계를 비교
- true or false 반환

연산	연산자	자료형	결과값
같다	a == b	정수형, 부동소수형	a가 b와 같으면 참 그렇지 않으면 거짓
다르다	a != b	정수형, 부동소수형	a와 b가 같지 않으면 참 그렇지 않으면 거짓
작다	a < b	정수형, 부동소수형	a가 b보다 작으면 참 그렇지 않으면 거짓
작거나 같다	a <= b	정수형, 부동소수형	a가 b보다 작거나 같으면 참 그렇지 않으면 거짓
크다	a > b	정수형, 부동소수형	a가 b보다 크면 참 그렇지 않으면 거짓
크거나 같다	a >= b	정수형, 부동소수형	a가 b보다 크거나 같으면 참 그렇지 않으면 거짓

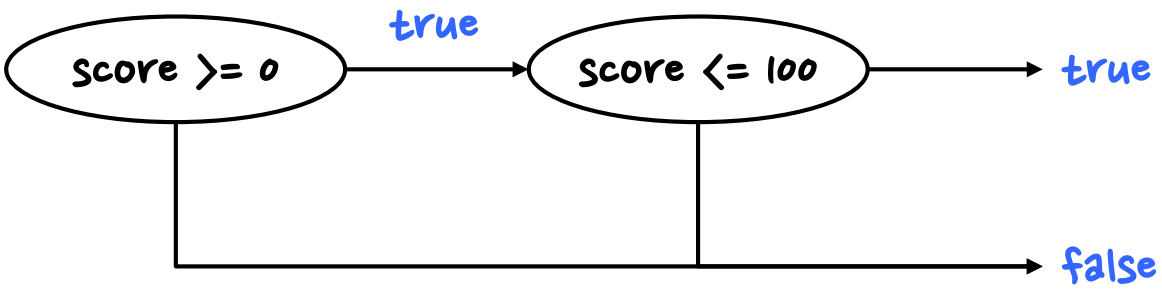
Logical operators (논리 연산자)

- true or false 반환

연산	연산자	결과값
논리 NOT	! a	a가 거짓이면 참 그렇지 않으면 거짓
논리 AND	a && b	a, b 모두 참이면 참 그렇지 않으면 거짓
논리 OR	a || b	a, b 중 적어도 한 개가 참이면 참 그렇지 않으면 거짓

Short circuit Evaluation of Logical Operators

- &&와 || 연산자는 short circuit evaluation 적용
- 왼쪽 피연산자가 연산의 결과값을 결정하는데 충분하면, 오른쪽 피연산자는 계산하지 않음
- e.g.
  - score >= 0 && score <= 100



Bit Operators (비트 연산자)

- Bit 단위 연산 결과값 반환

구분	연산	연산자	자료형	결과값
Bit shift operators	좌로 이동 (left shift)	a << n	정수형	a를 n 비트만큼 좌측으로 이동하고 오른쪽 끝은 이동한 수만큼 0으로 채운다
	우로 이동 (right shift)	a >> n	정수형	a를 n 비트만큼 우측으로 이동하고 왼쪽 끝은 이동한 수만큼 0 혹은 1로 채운다
	우로 이동 (right shift)	a >>> n	정수형	a를 n 비트만큼 우측으로 이동하고 왼쪽 끝은 이동한 수만큼 0으로 채운다
Bitwise logical operators	AND	a & b	정수형	a와 b의 비트별 AND 값
	OR	a | b	정수형	a와 b의 비트별 OR 값
	XOR	a ^ b	정수형	a와 b의 비트별 XOR 값
	1's complement	~a	정수형	a의 1의 보수 값

Bit operators

- Bit shift 활용

```
short i = 11; // 11 = 0000 0000 0000 1011
short j = 17; // 17 = 0000 0000 0001 0001
i = i << 2; // 0000 0000 0000 1011 << 2 = 0000 0000 0010 1100 = 44
j = j >> 3; // 0000 0000 0001 0001 >> 3 = 0000 0000 0000 0010 =  2
```

- Bitwise logical operators

연산	16진수 값	2진수 값
a	0x1f05	0001 1111 0000 0101
b	0x3121	0011 0001 1010 0001
~a	0xe0fa	1110 0000 1111 1010
a & b	0x1101	0001 0001 0000 0001
a | b	0x3fa5	0011 1111 1010 0101
a ^ b	0x2ea4	0010 1110 1010 0100

Assignment operators

- 변수에 값을 대입
  - a = a + 1;
    - "a와 a+1이 같다" (x)
    - "a+1을 a에 저장하라" (o)

- 대입 연산자 =
  - l-value에 대입한 값을 반환
  - 연속 대입
    - b = a = 2; $\Longrightarrow$ b = (a = 2);

Assignment operators

- 누적 대입 연산자
  - variable ◎= expression
  - ◎는 binary operator
  - variable = variable ◎ expression

```
a = a + 1;          a *= 1 + 3;
    ↓                   ↓
a += 1;             a = a * (1 + 3);
```

Incremental / Decremental Operator

- 증가 연산자(++)
  - 피연산자의 값을 하나 증가
  - a++; 는 a += 1;과 같은 의미
- 감소 연산자(--)
  - 피연산자의 값을 하나 감소
  - a--; 는 a -= 1;과 같은 의미

- 전치와 후치(prefix / postfix)
  - 증감 연산자는 전치, 후치에 따라 돌려주는 값이 달라진다
  - 전치일 경우(++a 혹은 --a)에는 증감 이후 변수 값을 돌려준다
  - 후치일 경우(a++ 혹은 a--)에는 증감 이전 변수 값을 돌려준다

Incremental / Decremental Operator

- Example
  - 초기 a, b 값이 5일 때
  - Prefix

단계	수행연산	수식	a 값	b 값
0	N/A	b = 2 * ++a	5	5
1	++	b = 2 * 6	6	5
2	*	b = 12	6	5
3	=	12	6	12

  - Postfix

단계	수행연산	수식	a 값	b 값
0	N/A	b = 2 * a++	5	5
1	++	b = 2 * 5	6	5
2	*	b = 10	6	5
3	=	10	6	10

conditional operator

- 유일한 ternary operator
  - condition ? first_expression : second_expression
  - condition이 참이면 first_expression, 거짓이면 second_expression의 계산 값 반환

- Example
  - a, b 중 큰 값 찾기
  - max = a > b ? a : b

Short circuit Evaluation of conditional operator

- condition의 true / false 여부에 따라 두 expression 중 하나만 연산

- Example
  - (n != 0) ? sum/n : 1;
  - Short circuit evaluation이 적용되지 않는다면,
    - n == 0 인 경우에도 sum/n 연산 수행
    - divide by zero 오류 발생

Parenthesis as operator

- 연산자가 아닌 괄호 ( )
  - 연산자와 피연산자의 관계를 나타내기 위한 괄호

    a = (b + c) * d;
- 연산자로 사용되는 괄호 ( )
  - 형 변환 연산에 사용된 괄호

    avg = sum / (double) n;

    avg = sum / ((double) n);

Precedence & Associativity

연산기호	결합방향	우선순위	
[ ], .	➡	1(높음)	
expr++, expr--	⬅	2	단항 연산
++expr, -- expr, +expr, -expr, ~, !, (type)	⬅	3	
*, /, %	➡	4	
+, -	➡	5	
〈〈, 〉〉, 〉〉〉	➡	6	
〈, 〉, 〈=, 〉=, instanceof	➡	7	
==, !=	➡	8	
&	➡	9	
^	➡	10	
|	➡	11	
&&	➡	12	
||	➡	13	
? expr : expr	⬅	14	조건 연산
=, +=, -=, *=, /=, %=, &=, ^=, |=, 〈〈=, 〉〉=, 〉〉〉=	⬅	15(낮음)	대입 연산

산술 → 관계 → 논리 → 대입

Summary

- 모든 연산자는 반환값이 있다
- 연산 종류에 따라
  - arithmetic / relational / logical / bit / assignment / incremental / decremental / conditional
- Precedence & Associativity
- Short circuit evaluation
  - logical / conditional operators
- Prefix / postfix
  - incremental / decremental operators

실습

- A = {(25+5)+(36/4)-72}*5
  B = {(25*5)+(36-4)+71}/4
  C = (128/4)*2 일 때
  A > B > C 이면 true, 그렇지 않으면 false를 출력

- 정수 36895의 오른쪽 첫번째 비트와 10번째 비트를 출력

- 초 단위 시간을 입력 받아 시, 분, 초의 표현으로 출력

```
초 값을 입력하세요
9961
9961 초는 2 시간 46 분 1 초입니다.
```

실습

- 근의 공식에 따라 이차방정식의 두 근을 계산
  - $ax^2 + bx + c = 0$ 의 계수 a, b, c를 입력 받아 이차방정식의 근 계산
  - Math 클래스 이용
    - Math.sqrt() : 제곱근

실습

- 체질량지수(Body Mass Index : BMI)를 계산하여 비만 여부를 판별

$$\text{BMI}(\text{몸무게}, \text{키}) = \frac{\text{몸무게} \times 9998}{\text{키}^2}$$

BMI	판정
20 미만	저체중
20 이상 25 미만	정상
25 이상 30 미만	과체중
30 이상	비만

```
몸무게를 입력하시오 : 70
키를 입력하시오 : 180
당신의 체질량 지수(BMI)는 21.600617
정상입니다.
```