Day-7

C++ Operators

연산자 우선순위

- 단항 연산자는 바이너리 연산자보다 우선 순위가 높다
- 표준 수학 연산자(+, *, etc...)는 비교, 비트 및 논리 연산자보다 우선 순위가 높다
- 비트 연산자 및 논리 연산자는 비교 연산자보다 우선 순위가 높다
- 복합 대입 연산자 (+=, -=, %=, etc...)는 우선 순위가 낮다
- 괄호()는 표현을 복잡하게 만들 때도 있지만 도움이 되는 경우도 있다

Prefix/Postfix Increment Semantic

- Prefix Incremnet/Decrement (++i, —i)
  1. 값을 업데이트 하고
  2. 업데이트 된 값을 반환함
- Postfix Increment/Decrement (i++, i—)
  1. 이전 값 저장(임시적)
  2. 값을 업데이트
  3. 업데이트 이전 값을 반환
- Prefix/Postfix Increment Semantic은 객체에도 적용됨

대입, 복합 및 쉼표(,) 연산자

- 대입 및 복합 연산자는 오른쪽에서 왼쪽으로 연관성을 가짐
- 쉼표 연산자는 왼쪽에서 오른쪽으로 연관성을 가짐
  - 왼쪽 식을 평가한 후 그 결과를 버리고 오른쪽 식을 반환함

Spaceship Operator <=> : 우주선 연산자

- C++20부터는 우주선 연산자라고 불리는 3방향 비교 연산자 <=>를 제공
- 두 개의 객체를 비교할 수 있음
- 이 연산자는 양수, 0 또는 음수 값과 직접 비교할 수 있는 객체를 반환

```
(3 <=> 5) == 0; // 3과 0은 동등하지 않기 때문에 false
('a' <=> 'a') == =; // 문자 'a'와 문자 'a'는 동등 true
(3 <=> 5) < 0;  // 3은 5보다 작기 때문에 true
```

안전 비교 연산자

- C++20은 서로 다른 유형(signed, unsigned)의 정수를 안전하게 비교하는 <utility> 함수 집합을 도입함

```
bool cmp_equl(T1 a, T2 b)
bool cmp_not_equal(T1 a, T2 b)
// etc.......
```

- 사용 예시

```
#include<utility>
unsigned a = 4;
int b= -3;
bool v1 = (a > b);                // false
bool v2 = std::cmp_greater(a, b) // true
```