**190408**

**\*Bitwise Operator**

<의의>

1. bool 자료형(8bit)의 경우 0 1 만 표현하고 나머지 공간은 놀기 때문에 빈 공간에 비트단위 연산자를 넣었다.

2. 비트단위 연산을 할 경우 계산 속도가 빠르다.

#include <bitset> : 2진수로 바꿔서 출력해주는 라이브러리

std::bitset<bit\_length>(Variable) //a(Variable)변수를 4(bit\_length)비트 2진수로 출력한다.

Tip) 비트 연산을 할 때 Unsigned를 쓰는 게 일반적.

**\*Bit Flag, Bit Mask**

//예시1. rpg게임상에서 아이템의 개수가 비트 수만큼 정해져 있고, 캐릭터에게 그 아이템을 가지고 있는지 검사를 할 때.

//예시2. SNS에서 게시물을 보거나, 편집했거나, 삭제했거나, 좋아요를 눌렀거나, 공유를 하거나 등의 여부를 검사할 때.

//기본 형태

1. 필요한 정보가 몇 개가 있는지 생각 해본다.

2. 만약 8개의 데이터가 필요하다면 8비트 무부호 자료형이 뭐가 있는지 생각해본다 -> unsigned char

3. 데이터를 담을 flag변수 하나와 특정 비트를 변환할 때 쓰일 Mask 변수(옵션)를 선언 후 초기화 한다.

Unsigned char My\_flag = 0;

unsigned char opt0 = 1 << 0;

Unsigned char opt1 = 1 << 1;

Unsigned char opt2 = 1 << 2;

Unsigned char opt3 = 1 << 3;

예시 1) 아이템 검사

//2^0번째 자리에 해당하는 아이템 정보의 유무를 검사할 때

If(My\_flag & opt0)

예시 2) 아이템 다중 검사

//2^1번째와 2^2번째 자리에 해당하는 아이템 정보의 유무를 검사

If((My\_flag & opt1) && (My\_flag & opt2))

예시 3) 상태 변환

//2^3번째 자리에 해당하는 아이템의 정보를 가지고 있으면 잃게 하고 싶을때

If(My\_flag & opt3) My\_flag ^= opt3;

* 비트마스크는 예제 참고.