C++의 기본 자료형

-> 자료형 자체로는 어떠한 의미가 없으며, 변수에 대한 타입을 설명하는 역할임

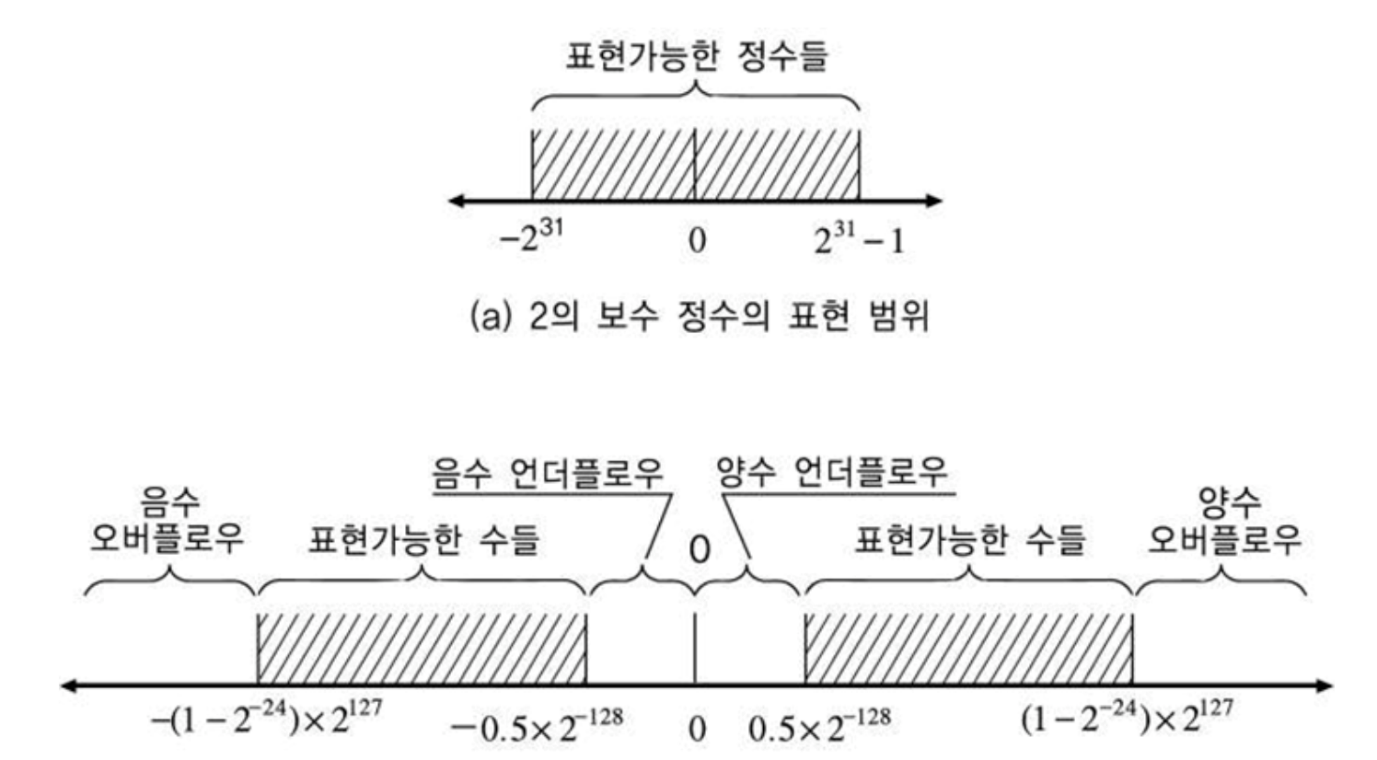
테이블이(가) 표시된 사진

자동 생성된 설명

\* unsigned / signed : 부호의 유무로, unsigned는 0부터 시작, signed는 2^-31부터 시작함.

-> 게임에서 예시 ) 몬스터 체력이 21억이 넘는다면, signed의 경우, 버그로 인해 갑자기 몬스터가 죽는 경우가 나타날 수 있음

정수 표현 범위



심볼릭 상수

- const 키워드와 함께 선언된 상수

Ex) const int ages = 30;

-> 상수의 타입을 명시적으로 지정 가능

-> 사용자 정의 타입에도 사용 가능 ex) 구조체

-> 해당 상수의 사용 범위를 개발자가 제한할 수 있음

Auto 키워드

- 변수를 초기화 할 때, 자동으로 초기값에 맞춰 타입을 선언해주는 키워드

C++ 타입 변환

- 다른 타입끼리의 대입, 산술연산 시 / 함수에 인수 전달 시 타입 변환 사용

- 묵시적 타입 변환(자동)

-> 컴파일러가 대입 연산이나 산술 연산에서 자동으로 수행

-> 대입 연산자 에서는 오른쪽 타입이 왼쪽으로 변환

-> 산술 연산에서는 데이터 손실이 적은 쪽으로 진행

- 명시적 타입 변환(강제)

-> 개발자가 Cast 연산자를 이용해 강제적으로 변환

-> (데이터 타입) 변수명 -> c와 c++에서 사용 / 데이터타입(변수명) -> c++에서만 사용

-> 타입캐스트 연산자 라고도 함

New 연산자 - 메모리 동적할당

- 사용 방법 : 타입\* 포인터이름 = new 타입;

-> new 연산자는 자유 기억 공간에 객체를 위한 메모리를 할당받게 함

-> new 연산자로 할당받은 메모리는 이름이 없어서, 포인터로만 접근 가능.

\*malloc과의 차이점

-> 별도의 자료형 선언이 필요없음

-> 초기값을 줄 수 있음(생성자 호출)

-> 에러 발생시 예외처리 실행, Null값을 반환하지 않음

-> realloc과 같은 메모리 크기 재조정이 불가능함

Delete 연산자

- 사용 방법 : delete 포인터이름;

-> new로 선언하지 않은 변수는 delete로 해제 불가능

\*free와의 차이점

-> 소멸자를 호출하지 않음

스마트 포인터

- unique\_ptr

이 포인터가 어떤 객체를 가지고 있을 때만, 소멸자가 객체를 삭제할 수 있음

-> 오직 하나의 객체만을 가리킬 수 있는 포인터로, move()함수를 통해 소유권을 이전하거나, 객체를 삭제시켜서 포인터를 해제하는 방법 밖에 없음 -> 안전한 포인팅 보장

텍스트이(가) 표시된 사진

자동 생성된 설명

+ make\_unique<타입>(데이터) 를 통해 유니크포인터를 위한 객체를 만들 수도 있음(c++14)



-> 유니크포인터 객체들은 모두 delete가 필요 없이, 컴파일러가 자동으로 해제시켜줌.

- shared\_ptr

특정 객체를 참조하는 포인터의 개수를 참조하는 포인터 -> 참조 횟수를 가리킴

-> 첫 생성 시에는 초기값이 1이며, 선언 할 때 마다 점점 늘어나고, 참조 횟수가 0이면 수명이 다한걸로 판단하여 메모리를 해제시킴

텍스트이(가) 표시된 사진

자동 생성된 설명

-> use\_count()함수를 통해 참조 횟수를 가져올 수 있음

-> 위의 그림의 경우에는, ptr01 자기자신만 있을때는 1, ptr02와 ptr03이 선언될 때 마다 1씩 증가하는 경우를 볼 수 있음.

텍스트이(가) 표시된 사진

자동 생성된 설명

-> make\_shared를 통해 앞의 make\_unique처럼 객체를 생성시켜줄 수 있음

-> 리턴타입은 객체가 아닌, 그 객체를 가리키는 포인터를 반환해줌.

- weak\_ptr

Shared\_ptr이 가리키는 객체에 접근할 수 있는 포인터지만, shared\_ptr의 참조 횟수에는 카운트가 되지 않는 포인터임.

-> shared\_ptr이 2개가 있고, 서로 상대방을 가리키고 있는 상태라면, 이 둘은 순환 참조가 되어 절대로 메모리에서 해제되지 않는 상태를 유지하게 됨(어느것을 먼저 해제해야 할 지 우선순위가 없는 경우)

-> shared\_ptr과 함께 사용하여, 순환 참조상태에 빠지지 않게 하는 역할.

-> shared\_ptr이나, 같은 weak\_ptr로만 복사, 대입이 가능함