constexpr을 사용하면 컴파일 타임에 상수임이 보장 되어야 함.

프로그래머가 기억해야 할 내용이 많을수록 버그가 생길 확률이 높다.

for range TS가 17사양에 좀더 들어올 예정이다.

for문을 for range문으로 변경해 보았다. 프로그래머가 범위를 지정하지 않아 실수를 줄일 수 있다. Boost의 range문을 사용하면 value와 index를 모두 사용 할 수 있다.

<http://www.boost.org/doc/libs/1_60_0/libs/range/doc/html/range/reference/adaptors/reference/filtered.html>

**decay란 무엇인가???**

기존에 가지고 있는 정보 중에서 누락되는 현상

void showValue(int nums[8]) // decay -> int\* num;이 되어 부식(decay) 됨.

{

For(auto&& item : nums){ // 여기에서 에러.

}

}

// 이렇게 바꾸면 nums를 레퍼런싱되기 때문에 아래 구문에서 에러가 발생하지 않음.

void showValue(int (&nums)[8])

{

For(auto&& item : nums){

}

}

template<typename T, std::size\_t N>

void showValue(T (&nums)[N])

{

For(auto&& item : nums){

}

}

Std::rank를 이용하면 배열의 차수를 알 수 있습니다.