객체지향프로그래밍

Hw03

**C211123 이준선**

**2023.03.25**

# 코드

(RAW FORMAT)

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <random>

/// @brief uniform distributed random even number generator

class EvenRandom

{

public:

/// @brief seed 설정

EvenRandom();

/// @brief 랜덤 짝수 리턴

/// @return random even number

int next();

/// @brief low와 high 사이의 랜덤 짝수 리턴

/// @param low minimum value

/// @param high maximum value

/// @return [low, high] 사이의 랜덤 짝수

int nextInRange(const int low, const int high);

private:

std::random\_device random\_device;

std::mt19937 mt\_engine;

std::uniform\_int\_distribution<int> uniform\_distributor;

};

EvenRandom::EvenRandom()

: mt\_engine(random\_device())

{

// 최대 분포를 RAND\_MAX의 절반으로 설정 (결과값에 \* 2를 하기 위함)

uniform\_distributor.param(std::uniform\_int\_distribution<int>::param\_type(0, RAND\_MAX / 2));

}

int EvenRandom::next()

{

return uniform\_distributor(mt\_engine) \* 2;

}

int EvenRandom::nextInRange(const int low, const int high)

{

std::uniform\_int\_distribution<int> ranged\_distributor(low / 2, high / 2);

return ranged\_distributor(mt\_engine) \* 2;

}

int main()

{

EvenRandom r;

std::cout << "-- 0에서 " << RAND\_MAX << "까지의 랜덤 짝수 정수 10 개--" << std::endl;

for (int i = 0; i < 10; i++)

{

int n = r.next(); // 0에서 RAND\_MAX(32767) 사이의 랜덤한 정수 cout << n << ' ';

std::cout << n << ' ';

}

std::cout << std::endl

<< std::endl

<< "-- 2에서 "

<< "10 까지의 랜덤 짝수 정수 10 개 --" << std::endl;

for (int i = 0; i < 10; i++)

{

int n = r.nextInRange(2, 10); // 2에서 10 사이의 랜덤한 정수 cout << n << ' ';

std::cout << n << ' ';

}

std::cout << std::endl;

}

(STYLED FORMAT)

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <random>

/// @brief uniform distributed random even number generator

class EvenRandom

{

public:

/// @brief seed 설정

EvenRandom();

/// @brief 랜덤 짝수 리턴

/// @return random even number

int next();

/// @brief low와 high 사이의 랜덤 짝수 리턴

/// @param low minimum value

/// @param high maximum value

/// @return [low, high] 사이의 랜덤 짝수

int nextInRange(const int low, const int high);

private:

std::random\_device random\_device;

std::mt19937 mt\_engine;

std::uniform\_int\_distribution<int> uniform\_distributor;

};

EvenRandom::EvenRandom()

: mt\_engine(random\_device())

{

// 최대 분포를 RAND\_MAX의 절반으로 설정 (결과값에 \* 2를 하기 위함)

uniform\_distributor.param(std::uniform\_int\_distribution<int>::param\_type(0, RAND\_MAX / 2));

}

int EvenRandom::next()

{

return uniform\_distributor(mt\_engine) \* 2;

}

int EvenRandom::nextInRange(const int low, const int high)

{

std::uniform\_int\_distribution<int> ranged\_distributor(low / 2, high / 2);

return ranged\_distributor(mt\_engine) \* 2;

}

int main()

{

EvenRandom r;

std::cout << "-- 0에서 " << RAND\_MAX << "까지의 랜덤 짝수 정수 10 개--" << std::endl;

for (int i = 0; i < 10; i++)

{

int n = r.next(); // 0에서 RAND\_MAX(32767) 사이의 랜덤한 정수 cout << n << ' ';

std::cout << n << ' ';

}

std::cout << std::endl

<< std::endl

<< "-- 2에서 "

<< "10 까지의 랜덤 짝수 정수 10 개 --" << std::endl;

for (int i = 0; i < 10; i++)

{

int n = r.nextInRange(2, 10); // 2에서 10 사이의 랜덤한 정수 cout << n << ' ';

std::cout << n << ' ';

}

std::cout << std::endl;

}

# 결과

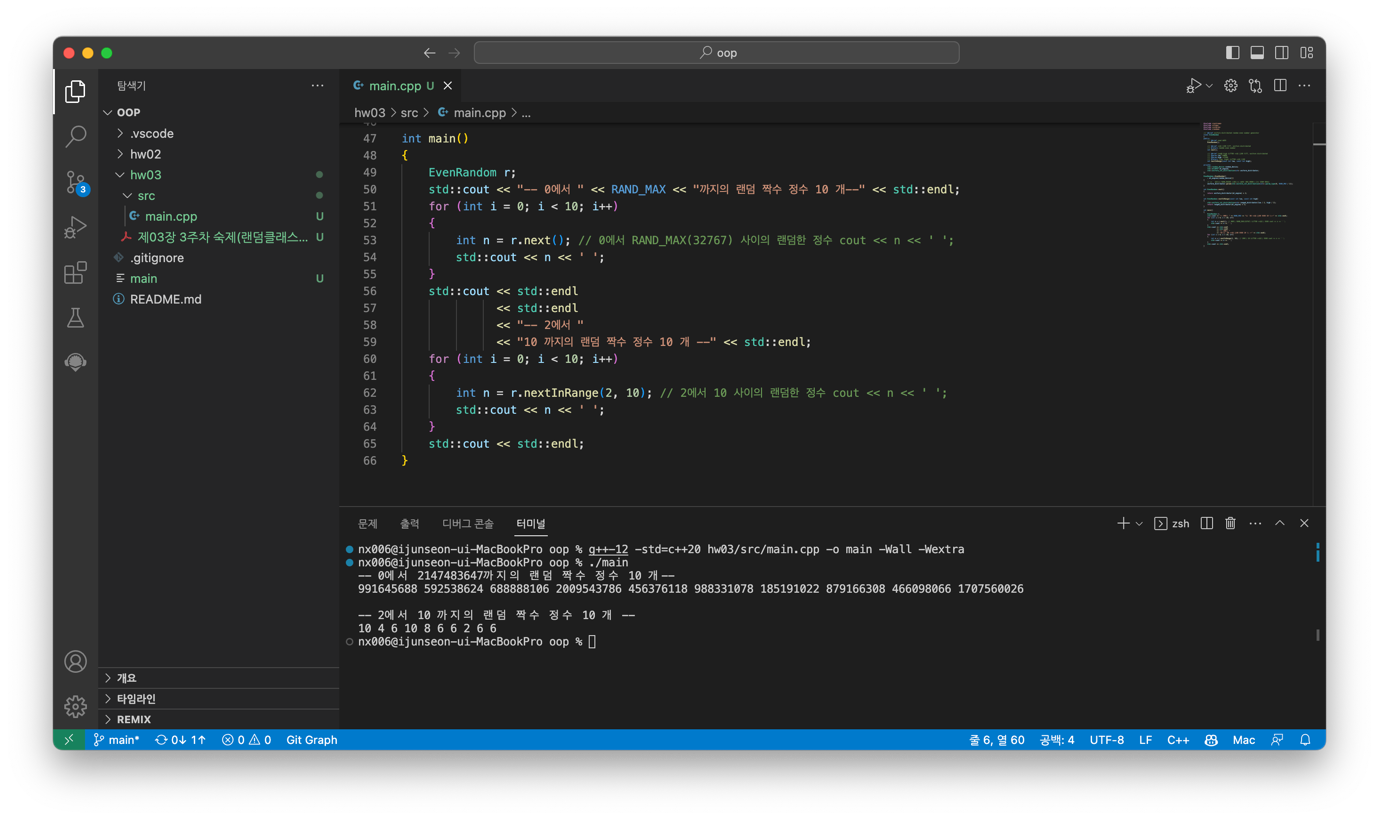


그림 1 실행 결과

텍스트, 모니터, 스크린샷, 화면이(가) 표시된 사진

자동 생성된 설명

그림 2 clang result

# 참고사항

* random 헤더는 C++11부터 도입됐기 때문에 컴파일러 옵션에 -std=c++11, 혹은 그 이상의 버전을 붙여주어야 한다.
* 그림 1에서 최댓값이 2147483647로 설정됐다. 과제 명세서에 적힌 값(32767)과 다른데, 컴파일러 혹은 C++ 버전에 따라 RAND\_MAX의 값은 달라질 것이므로 이를 굳이 수정하지는 않았다.
* C++버전은 C++20, 컴파일러는 g++-12 버전을 사용했다.
* 사실 컴파일러를 g++이 아닌 clang으로 바꾸고, C++ 버전을 11, 14, 17로 바꾸어도 과제 명세서에 적힌 값은 얻을 수 없었다(그림 2). 이는 애초에 cstdlib 에서 매크로로 정의된 RAND\_MAX의 값이 0x7fffffff로 int 범위의 최댓값(약 21억)으로 정의됐기 때문이다.