객체지향프로그래밍  
HW04 동적 메모리

C211123 이준선

2023.03.31

# 코드 (RAW Format)

#include <iostream>

class Sample

{

public:

Sample(const int n) : capacity(n), size(0)

{

p = new int[n];

}

void read();

void write() const;

int big();

int getSize() const

{

return size;

}

~Sample()

{

delete[] p;

}

private:

int \*p;

int capacity;

int size;

};

void Sample::read()

{

std::cout << "입력할려는 정수의 갯수는 ? ";

auto inputSize = 0;

std::cin >> inputSize;

// reallocate memory

// for optimization, capacity is double of inputSize

if (inputSize > capacity) {

capacity = 2 \* inputSize;

int \*temp = new int[capacity];

std::copy(p, p + this->size, temp);

delete[] p;

p = temp;

}

size = inputSize;

std::cout << inputSize << "개의 정수를 입력하시오 ";

for (auto idx = 0; idx < inputSize; idx++)

{

std::cin >> p[idx];

}

}

void Sample::write() const

{

for (auto idx = 0; idx < size; idx++)

{

std::cout << p[idx] << ' ';

}

std::cout << std::endl;

}

int Sample::big()

{

if (size == 0)

{

throw std::runtime\_error("size is 0");

}

auto firstIterator = p;

const auto lastIterator = p + size;

if (firstIterator != lastIterator)

{

auto currentIterator = firstIterator;

while (++currentIterator != lastIterator)

{

if (\*firstIterator < \*currentIterator)

{

firstIterator = currentIterator;

}

}

}

return \*firstIterator;

}

int main()

{

Sample s(10);

s.read();

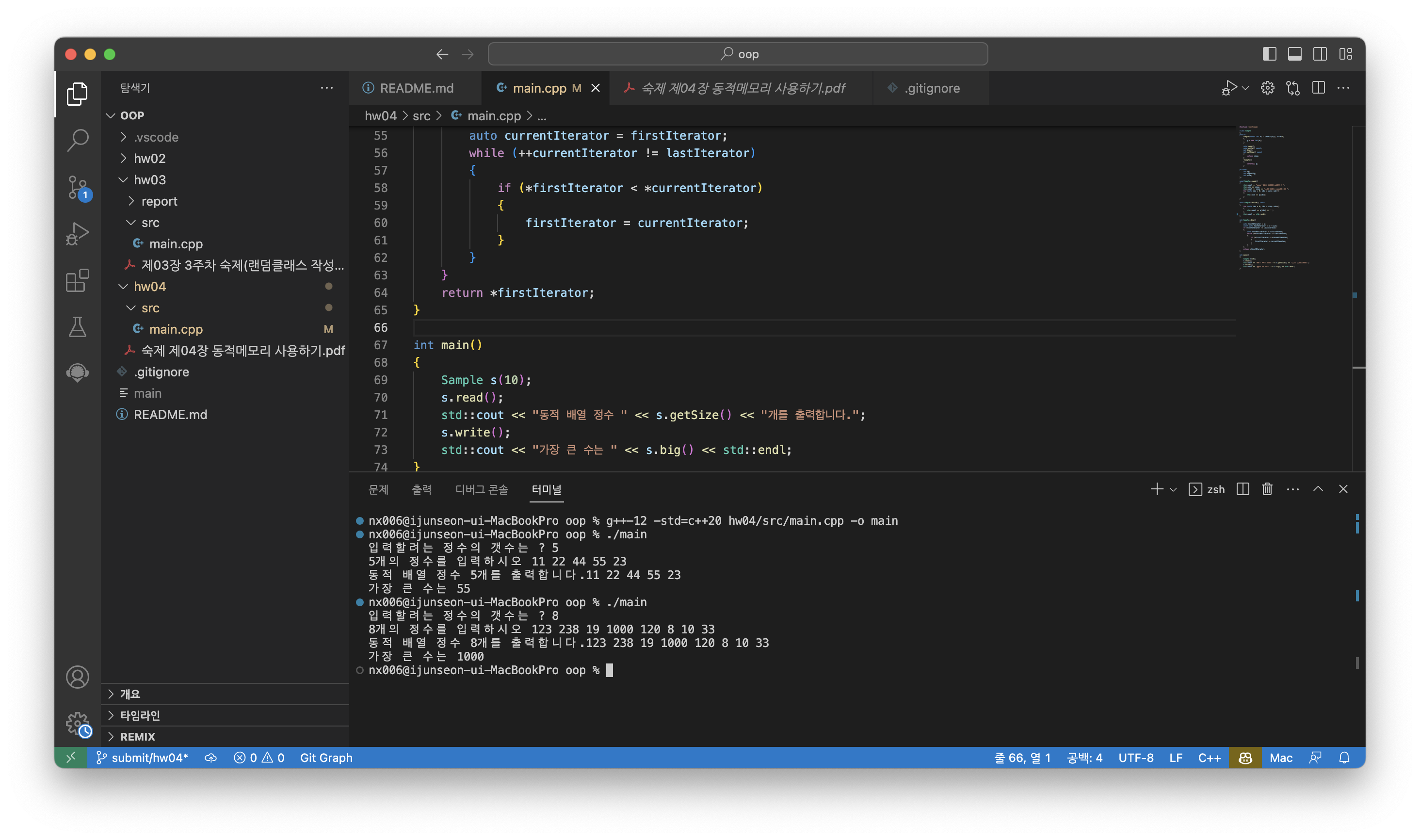
std::cout << "동적 배열 정수 " << s.getSize() << "개를 출력합니다.";

s.write();

std::cout << "가장 큰 수는 " << s.big() << std::endl;

}

# 실행 결과



# 소스코드 (Raw format)

#include <iostream>

class Sample

{

public:

Sample(const int n) : capacity(n), size(0)

{

p = new int[n];

}

void read();

void write() const;

int big();

int getSize() const

{

return size;

}

~Sample()

{

delete[] p;

}

private:

int \*p;

int capacity;

int size;

};

void Sample::read()

{

std::cout << "입력할려는 정수의 갯수는 ? ";

auto inputSize = 0;

std::cin >> inputSize;

// reallocate memory

// for optimization, capacity is double of inputSize

if (inputSize > capacity) {

capacity = 2 \* inputSize;

int \*temp = new int[capacity];

std::copy(p, p + this->size, temp);

delete[] p;

p = temp;

}

size = inputSize;

std::cout << inputSize << "개의 정수를 입력하시오 ";

for (auto idx = 0; idx < inputSize; idx++)

{

std::cin >> p[idx];

}

}

void Sample::write() const

{

for (auto idx = 0; idx < size; idx++)

{

std::cout << p[idx] << ' ';

}

std::cout << std::endl;

}

int Sample::big()

{

if (size == 0)

{

throw std::runtime\_error("size is 0");

}

auto firstIterator = p;

const auto lastIterator = p + size;

if (firstIterator != lastIterator)

{

auto currentIterator = firstIterator;

while (++currentIterator != lastIterator)

{

if (\*firstIterator < \*currentIterator)

{

firstIterator = currentIterator;

}

}

}

return \*firstIterator;

}

int main()

{

Sample s(10);

s.read();

std::cout << "동적 배열 정수 " << s.getSize() << "개를 출력합니다.";

s.write();

std::cout << "가장 큰 수는 " << s.big() << std::endl;

}