ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
О ПРАКТИЧЕСКОЙ РАБОТЕ № 2

По дисциплине «Языки программирования»

Выполнил: ст. гр. ТКИ – 241

Омельченко А.А.

Проверил: к.т.н., доц.

Васильева М. А.

Москва 2023

Задание 1

## Формулировка задачи

Реализовать задание 4.3 из прошлого семестра в ООП стиле.

Создать решение (Solution), которое минимально содержит три проекта (Projects): исполняемый (Console Application), библиотеку классов (Library), и модульные тесты (Tests). Разработать библиотеку классов по заданному варианту. Важно! Библиотека классов не должна зависеть от потоков ввода-вывода. Каждый класс необходимо размещать в отдельных *двух* файлах, снабжённых «говорящим именем» и специальными расширениями: \*.h для заголовочных файлов (Header), содержащих API класса, и \*.cpp для компилируемых (Source), содержащих реализацию класса. В запускаемом проекте требуется создать файл main.cpp, содержащий точку входа в демонстрационную программу – главную функцию (main). В рамках данной функции показать работу с массивом.

Создать класс Matrix (двумерный массив) (можно использовать std::vector<T>). Реализовать все конструкторы, создаваемые компилятором по умолчанию, реализовать деструктор. Предусмотреть методы вывода в строку содержимого массива. Переопределить операторы присваивания, сдвига влево и сдвига вправо. Переопределить оператор разыменования элемента коллекции по индексу.

Предусмотреть заполнение массива по определенному алгоритму (случайным образом, вводом с клавиатуры, заполнение нулями и константным значением) реализовать через класс Generator, конкретный алгоритм реализовать в классе наследнике.

## UML - диаграмма алгоритма

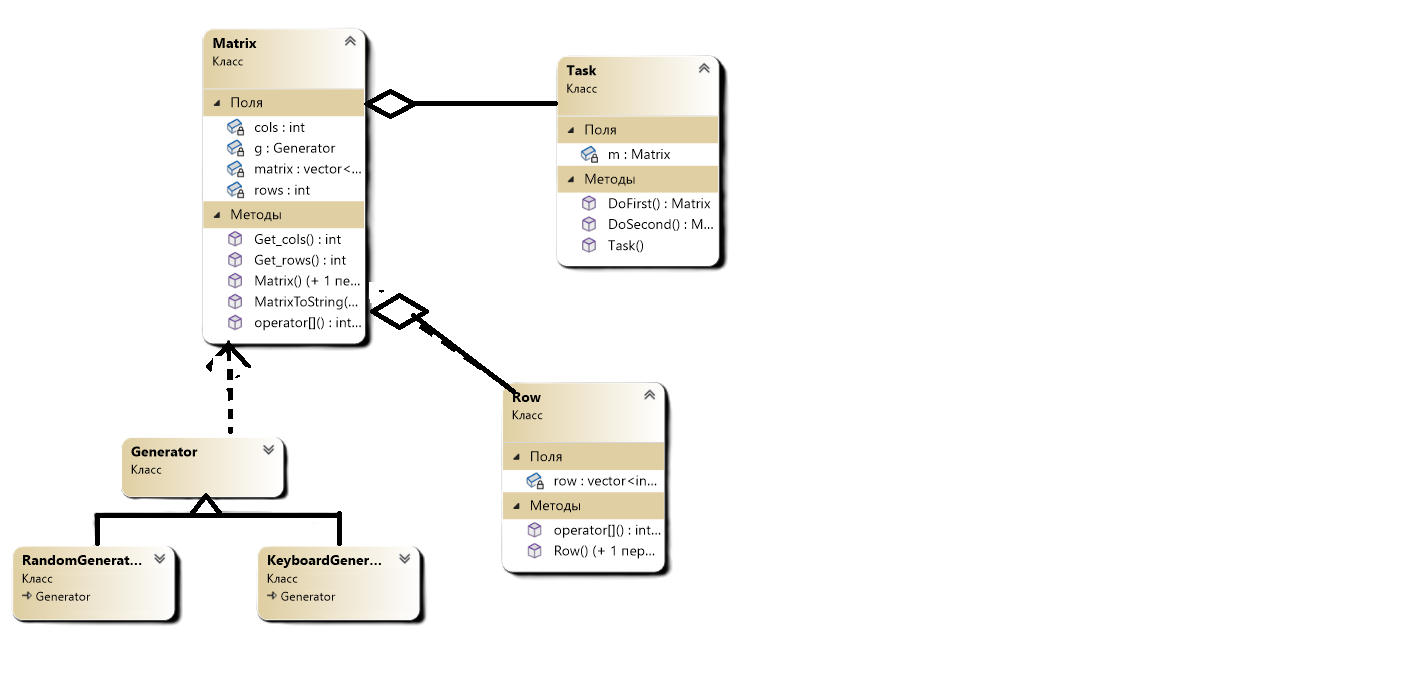


Рисунок 2 – UML диаграмма класса.

## Решение задачи на языке программирования C++

main.cpp:

#include <vector>

#include <string>

#include <iostream>

#include "..//Matrix/Matrix.h"

#include "..//Matrix/RandomGenerator.h"

#include "..//Matrix/Task.h"

using namespace matrix;

int main() {

setlocale(LC\_ALL, "rus");

enum class fillingarray {

c\_Random,

c\_Manual

};

int rows, cols;

std::cout << "enter row number ";

std::cin >> rows;

std::cout << "enter col number ";

std::cin >> cols;

int input = 0;

std::cout << "Выберите способ заполнения массива (Рандом = " << static\_cast<int>(fillingarray::c\_Random) << ", с клавиатуры = " << static\_cast<int>(fillingarray::c\_Manual) << ") " << "\n";

std::cin >> input;

const auto choise = static\_cast<fillingarray>(input);

switch (choise) {

case(fillingarray::c\_Random):

{

RandomGenerator gen(0, 100);

matrix::Matrix m = Matrix(rows, cols, &gen);

std::cout << m << std::endl << std::endl;

Task t = Task(m);

m =t.DoFirst();

std::cout << m << std::endl << std::endl;

m = t.DoSecond();

std::cout << m;

break; }

default:

std::cout << "Error! Нет такой функции";

return 1;

}

}

Matrix.h:

#pragma once

#include "Row.h"

#include "Generate.h"

#include <iostream>

namespace matrix {

class Matrix {

public:

Matrix(int rows, int cols, Generator\* g);

Matrix(int rows, int cols);

int Get\_rows();

int Get\_cols();

friend std::ostream& operator<<(std::ostream& os, const Matrix& mat);

Row& operator[](const int index);

std::string MatrixToString() ;

private:

std::vector<Row>matrix;

size\_t rows;

size\_t cols;

Generator\* g;

};

}

Matrix.cpp:

#include "Matrix.h"

#include <vector>

#include <string>

#include<sstream>

matrix::Matrix::Matrix(int rows,int cols) :

rows(rows), cols(cols), matrix(rows, Row(cols))

{

}

matrix::Matrix::Matrix(int rows, int cols, Generator\* g):

rows(rows), cols(cols),g(g), matrix(rows,Row(cols))

{

if (rows <= 0 or cols <= 0) {

throw std::out\_of\_range("Incorrect size");

}

else if (g == nullptr) {

throw std::out\_of\_range("Empty Gen");

}

else{

for (size\_t i = 0; i < rows; i++) {

for (size\_t j = 0; j < cols; j++) {

matrix[i][j] = g->generate();

}

}

}

}

Row& matrix::Matrix::operator[](const int index)

{

if (index < matrix.size()) {

return matrix[index];

}

else { throw std::out\_of\_range("Incorrect size"); }

}

int matrix::Matrix::Get\_rows()

{

return rows;

}

int matrix::Matrix::Get\_cols()

{

return cols;

}

std::string matrix::Matrix::MatrixToString()

{

std::stringstream buffer;

for (size\_t i = 0; i < rows; i++)

{

for (size\_t j = 0; j < cols; j++)

{

buffer << matrix[i][j] << ' ';

}

buffer << std::endl;

}

return buffer.str();

}

std::ostream& matrix::operator<<(std::ostream& os, const Matrix& mat)

{

for (const Row& row : mat.matrix) {

os << row << '\n';

}

return os;

}

Task.h:

#pragma once

#include <iostream>

#include "..//Matrix/Matrix.h"

class Task {

public:

Task(matrix::Matrix m);

matrix::Matrix DoFirst();

matrix::Matrix DoSecond();

private:

matrix::Matrix m;

};

Task.cpp:

#include "Task.h"

#include <cmath>

#include <vector>

Task::Task(matrix::Matrix m):

m(m)

{

}

matrix::Matrix Task::DoFirst()

{

int cols = m.Get\_cols();

int rows = m.Get\_rows();

for (int c = 0; c < cols; c++) {

int min = m[0][c];

int minR = 0;

for (int r = 0; r < rows; r++) {

if (abs(m[r][c]) < min) {

min = abs(m[r][c]);

minR = r;

}

}

m[minR][c] = 0;

}

return m;

}

matrix::Matrix Task::DoSecond()

{

int cols = m.Get\_cols();

int rows = m.Get\_rows();

std::vector<int> a;

for (int c = 0; c < cols; c++) {

if (m[0][c] <= m[rows - 1][c]) {

a.push\_back(c);

}

}

int cols1 = a.size();

matrix::Matrix m1 = matrix::Matrix(rows,cols1);

int col\_i = 0;

for (col\_i = 0; col\_i < cols1; col\_i++) {

for (int r = 0; r < rows; r++) {

m1[r][col\_i] = m[r][a[col\_i]];

}

}

return m1;

}

Generator.h:

#pragma once

namespace miit::algebra

{

/\*\*

\* @brief Абстрактный класс Generator.

\*/

class Generator

{

public:

/\*\*

\* @brief Виртуальный деструктор.

\*/

virtual ~Generator() = 0 {};

/\*\*

\* @brief Виртуальный метод генератор.

\*/

virtual int generate() = 0;

};

}

KeyboardGenerator.h:

#pragma once

#include"Generate.h"

class KeyboardGenerator : public Generator {

private:

std::istream& in;

public:

KeyboardGenerator(std::istream& in = std::cin);

int generate() override;

};

IStreamGenerator.cpp:

#include "KeyboardGenerator.h"

KeyboardGenerator::KeyboardGenerator(std::istream& in):

in(in)

{

}

int KeyboardGenerator::generate()

{

int value = 0;

this->in >> value;

return value;

}

RandomGenerator.h:

#pragma once

#include <random>

#include"Generate.h"

class RandomGenerator : public Generator {

private:

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

std::mt19937 generator;

public:

RandomGenerator(const int min, const int max);

// Реализация функции генерации случайного числа

int generate() override;

};

RandomGenerator.cpp:

#include "RandomGenerator.h"

RandomGenerator::RandomGenerator(const int min, const int max)

{

this->generator = std::mt19937(std::random\_device{}());

this->distribution = std::uniform\_int\_distribution<int>(min, max);

}

int RandomGenerator::generate()

{

return this->distribution(this->generator);

}

## Решение тестов

Код тестов:

#include "pch.h"

#include "CppUnitTest.h"

#include "../Matrix/RandomGenerator.h"

#include "../Matrix/Matrix.h"

#include"../Matrix/Task.h"

#include"../Matrix/Row.h"

#include "../Matrix/KeyboardGenerator.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

using namespace matrix;

namespace MatrixTest

{

TEST\_CLASS(MatrixTest)

{

public:

TEST\_METHOD(ConstructorTest\_ValidData\_Success)

{

RandomGenerator gen(-10, 10);

Matrix m(3, 3, &gen);

Assert::IsNotNull(&m);

}

TEST\_METHOD(Copy\_Test\_Success) {

RandomGenerator rnd(-10, 10);

Matrix matrix(3, 3, &rnd);

Matrix matrixtwo(matrix);

Assert::AreEqual(matrix.MatrixToString(), matrixtwo.MatrixToString());

}

TEST\_METHOD(GetCol\_Test\_Validdata\_Succes) {

RandomGenerator rnd(-10, 10);

Matrix matrix(3, 5, &rnd);

int expcol = 5;

Assert::AreEqual(matrix.Get\_cols(), expcol);

}

TEST\_METHOD(GetRow\_Test\_Validdata\_Succes) {

RandomGenerator rnd(-10, 10);

Matrix matrix(3, 5, &rnd);

int expcol = 3;

Assert::AreEqual(matrix.Get\_rows(), expcol);

}

};

TEST\_CLASS(TaskTest)

{

public:

TEST\_METHOD(ConstructorTest\_ValidData\_Success)

{

RandomGenerator rnd(-5, 5);

Matrix matrix(3, 3, &rnd);

Task task(matrix);

Assert::IsNotNull(&task);

}

TEST\_METHOD(Task\_Test\_DoFirst)

{

std::istringstream in("1 2 3 7 -5 2 4 1 8");

KeyboardGenerator key(in);

Matrix matrix(3, 3, &key);

Task task(matrix);

std::string expstr = "0 2 3 \n7 -5 0 \n4 0 8 \n";

matrix = task.DoFirst();

Assert::AreEqual(matrix.MatrixToString(),expstr);

}

TEST\_METHOD(Task\_Test\_DoSecond)

{

std::istringstream in("0 2 3 7 -5 0 4 0 8");

KeyboardGenerator key(in);

Matrix matrix(3, 3, &key);

Task task(matrix);

std::string expstr = "0 3 \n7 0 \n4 8 \n";

matrix = task.DoSecond();

Assert::AreEqual(matrix.MatrixToString(), expstr);

}

};

}

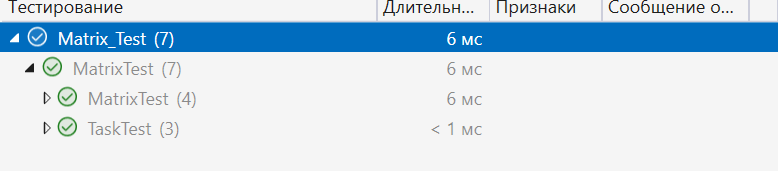


Рисунок 4 – Успешные тесты.

## Зачет задания в GitHub

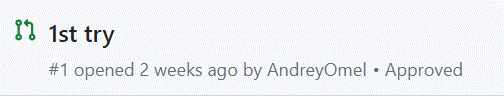


Рисунок 5 - Зачет задания в GitHub