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

Хугаева М. М.

Проверил: к.т.н., доц.

Васильева М. А.

Москва 2023

Код:

#pragma once

#include "/Users/yanmi/source/repos/Matrix/Matrix/Generator/Generator.h"

#include <iostream>

#include <ostream>

#include <sstream>

#include <vector>

/\*\*

\* @brief Класс матрицы

\*/

class Matrix

{

public:

/\*\*

\* @brief Создание объекта типа матрицы с введенными параметрами

\* @param rows количество строк матрицы

\* @param collums количество столбцов матрицы

\* @param Generator способ заполнения матрицы

\*/

Matrix(int rows, int collumns, Generator\* generator);

Matrix(std::vector<std::vector<int>> data);

/\*\*

\* @brief Количество строк в матрице

\*/

size\_t rowsQuantity() const;

/\*\*

\* @brief Количество столбцов в матрице

\*/

size\_t collumnsQuantity() const;

/\*\*

\* @brief Перегрузка оператора вывода

\* @param os Вывод

\* @param matrix Матрица, откуда берются данные для вывода

\* @return Вывод

\*/

friend std::ostream& operator<<(std::ostream& os, Matrix& matrix) noexcept;

/\*\*

\* @brief Перегрузка оператора сравнения

\* @param lha Первый аргумент для сравнения

\* @param rha Второй аргумент для сравнения

\*/

friend bool operator==(Matrix& lha, Matrix& rha) noexcept;

/\*\*

\* @brief Перегрузка оператора квадратные скобки

\* @param index Индекс элемента в массиве

\* @return Элемент из матрцы стоящий под выбранным индексом

\*/

std::vector<int>& operator[](size\_t index);

/\*\*

\* @brief Преобразование объекта матрицы в строку

\* @return Строка с информацией о матрице

\*/

std::string toString() const noexcept;

/\*\*

\*@ brief Получение информации

\*/

std::vector<std::vector<int>> getData();

private:

std::vector<std::vector<int>> data;

};

#include "Matrix.h"

#include <random>

#include <sstream>

#include <vector>

Matrix::Matrix(int rows, int collumns, Generator\* generator):

data({})

{

if (rows <= 0 || collumns <= 0)

{

throw std::logic\_error("Размер должен быть положительным");

}

if (generator == nullptr)

{

throw std::logic\_error("Некорректные данные");

}

rows = static\_cast<size\_t>(rows);

collumns = static\_cast<size\_t>(collumns);

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

{

std::vector<int> temp{};

for (size\_t c = 0; c < collumns; c++)

{

temp.push\_back(generator->generate());

}

this->data.push\_back(temp);

}

};

Matrix::Matrix(std::vector<std::vector<int>> data)

{

this->data = data;

}

size\_t Matrix::Matrix::rowsQuantity() const

{

return this->data.size();

}

size\_t Matrix::Matrix::collumnsQuantity() const

{

if (this->data.size() == 0)

{

return 0;

}

return this->data[0].size();

}

std::vector<int>& Matrix::operator[](size\_t index)

{

return data[index];

}

std::string Matrix::toString() const noexcept

{

std::stringstream temp;

for (size\_t s = 0; s < this->rowsQuantity(); s++)

{

for (size\_t c = 0; c < this->collumnsQuantity(); c++)

{

temp << this->data[s][c] << ' ';

}

temp << "\n";

}

return temp.str();

}

std::vector<std::vector<int>> Matrix::getData()

{

return data;

}

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

{

return os << matrix.toString();

}

bool operator==(Matrix& lha, Matrix& rha) noexcept

{

return lha.toString() == rha.toString();

}

#pragma once

#include "MainExercise.h"

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

class Exercise : public MainExercise

{

private:

Matrix matrix;

public:

Exercise(Matrix matrix);

/\*\*

\* @brief Выполнение упражнения номер 1

\* @return Результат выполнения 1 упражнения

\*/

Matrix ExerciseOne() override;

/\*\*

\* @brief Выполнение упражнения номер 2

\* @return Результат выполнения 2 упражнения

\*/

Matrix ExerciseTwo() override;

/\*\*

\* @brief Проверка элемента в строке в матрице

\*/

bool isElementInRow(int value, size\_t rowId);

/\*\*

\* @brief Добавляет строку в матрицу по индексу

\* @param rowLine Строка для добавления

\* @param index Индекс для добавления строки

\*/

void addRow(std::vector<int> data);

};

#include "/Users/yanmi/source/repos/Matrix/Matrix/Matrix/Matrix.h"

#include "Exercise.h"

#include "MainExercise.h"

#include "/Users/yanmi/source/repos/Matrix/Matrix/Matrix/Matrix.h"

#include <vector>

Exercise::Exercise(Matrix matrix) :

matrix(matrix)

{

};

void Exercise::addRow(std::vector<int> datafirst)

{

std::vector<std::vector<int>> data = matrix.getData();

data.push\_back(datafirst);

};

bool Exercise::isElementInRow(int value, size\_t rowId)

{

std::vector<std::vector<int>> data = matrix.getData();

for (size\_t i = 0; i < data[rowId].size(); i++)

{

if (data[rowId][i] == value)

{

return true;

}

}

return false;

};

Matrix Exercise::ExerciseOne()

{

Matrix result{this->matrix};

for (size\_t c = 0; c <matrix.collumnsQuantity(); ++c)

{

std::vector<std::vector<int>> data = matrix.getData();

int min = std::numeric\_limits<int>::max();

for (size\_t s = 0; s < data.size(); ++s)

{

if ((data[s][c]) <= min)

{

min = data[s][c];

}

}

for (size\_t s = 0; s < matrix.rowsQuantity(); ++s)

{

if (result[s][c] == min)

{

{

result[s][c] = 0;

}

}

}

}

return result;

};

Matrix Exercise::ExerciseTwo()

{

std::vector<std::vector<int>> data = matrix.getData();

size\_t rows = data.size();

size\_t columns = data[0].size();

std::vector<int> zeroLine(columns);

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

{

if (matrix[i][0] % 3 == 0)

{

data.insert(data.begin() + i, zeroLine);

}

}

Matrix matrixTwo = Matrix(data);

return matrixTwo;

}

#pragma once

#include "/Users/yanmi/source/repos/Matrix/Matrix/Matrix/Matrix.h"

#include "Exercise.h"

#include <vector>

class MainExercise

{

public:

virtual ~MainExercise() = 0 {};

virtual Matrix ExerciseOne() = 0;

virtual Matrix ExerciseTwo() = 0;

};

#include <iostream>

#include "../Generator/Generator.h"

#include "../Generator/RandomGenerator.h"

#include "../Generator/IStreamGenerator.h"

#include "../Exercise/Exercise.h"

#include "../Exercise/MainExercise.h"

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

int main()

{

setlocale(LC\_ALL, "Russian");

Generator\* generator = new RandomGenerator(0, 10);

Matrix matrix(4, 3, generator);

std::cout << matrix << std::endl;

std::cout << "---------------------------------" << std::endl;

Exercise Exercise = matrix;

std::cout << Exercise.ExerciseOne().toString();

std::cout << "---------------------------------" << std::endl;

std::cout << Exercise.ExerciseTwo().toString();

return 0;

}

Тесты:

#include "pch.h"

#include "CppUnitTest.h"

#include "/Users/yanmi/source/repos/Matrix/Matrix/Matrix/Matrix.h"

#include "/Users/yanmi/source/repos/Matrix/Matrix/Generator/IStreamGenerator.h"

#include "../Generator/RandomGenerator.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace TestMatrix

{

TEST\_CLASS(TestMatrix)

{

public:

TEST\_METHOD(MatrixToString\_ValidData\_Success)

{

Generator\* generator = new RandomGenerator(0, 10);

Matrix matrix(3, 2, generator);

Assert::IsFalse(matrix.toString() == "");

};

TEST\_METHOD(MatrixRowsCounts\_ValidData\_Success)

{

Generator\* generator = new RandomGenerator(0, 10);

Matrix matrix{ 3, 3, generator };

int temp = matrix.rowsQuantity();

Assert::IsTrue(temp == 3);

}

TEST\_METHOD(MatrixCollumnssCounts\_ValidData\_Success)

{

Generator\* generator = new RandomGenerator(0, 10);

Matrix matrix{ 3, 3, generator };

int temp = matrix.collumnsQuantity();

Assert::IsTrue(temp == 3);

}

};

}

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 1 – Выполнение тестов

UML-диаграмма:

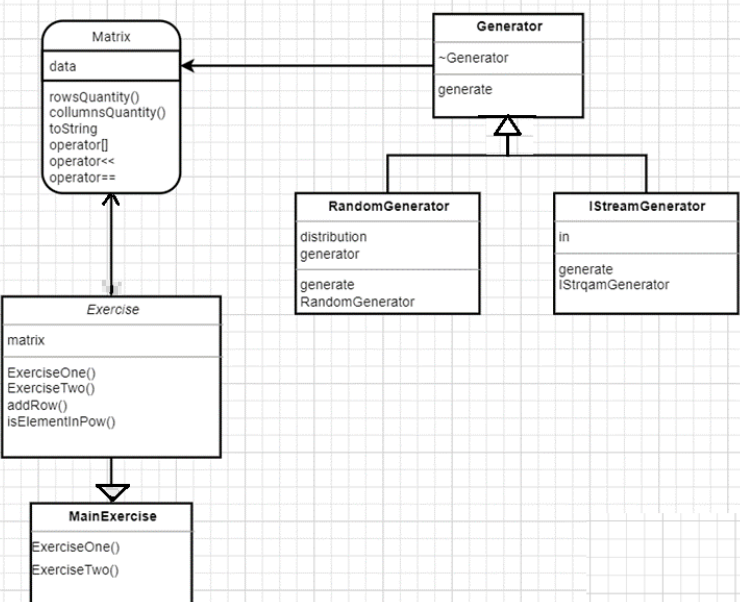




Рисунок 2 – UML-диаграмма

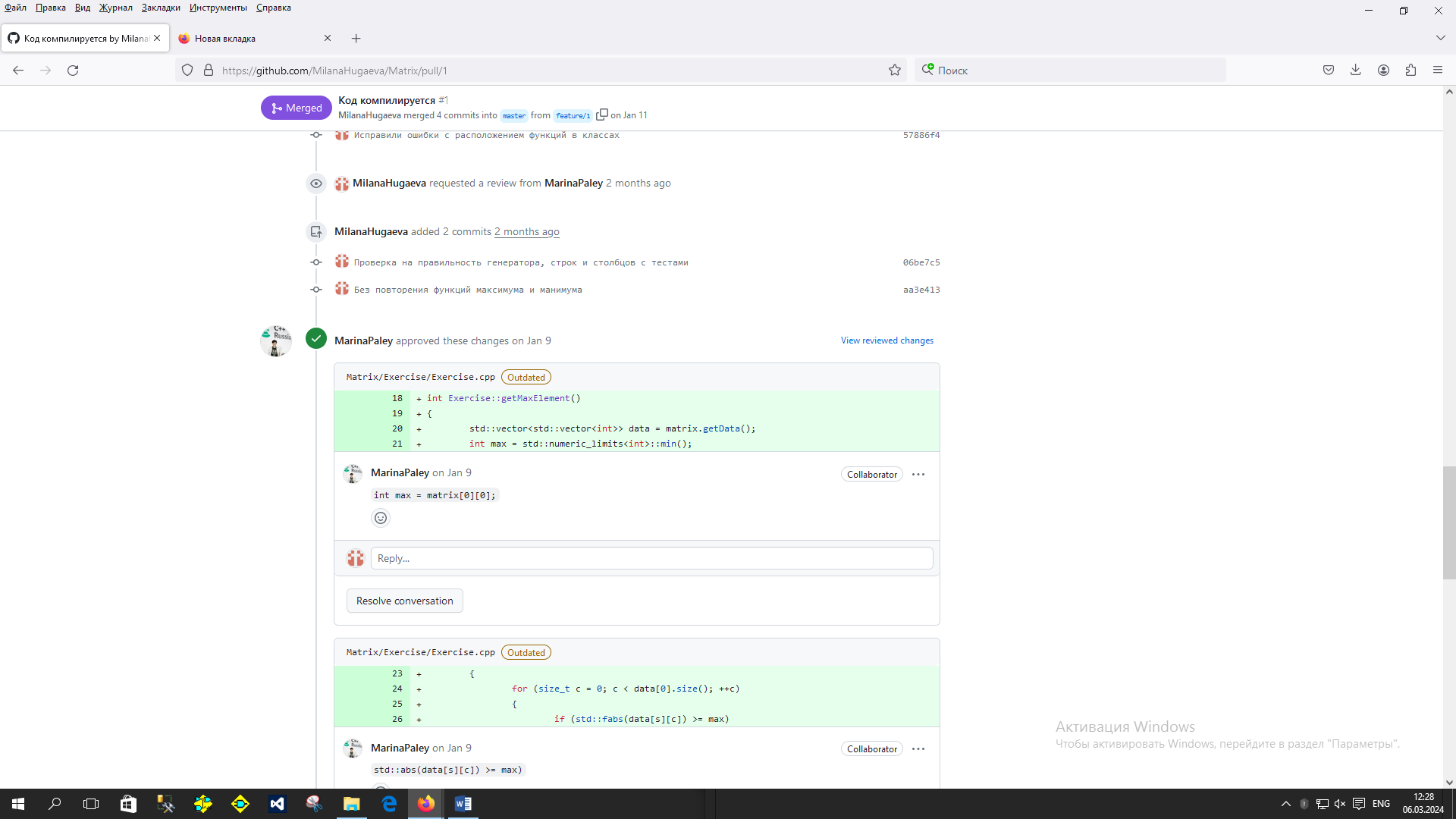


Рисунок 3 – Аппрув по заданию