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 № 1

По дисциплине «Языки программирования»

Выполнил: ст. гр. ТКИ – 241

Белавинская М.А.

Проверил: к.т.н., доц.

Васильева М. А.

Москва 2023

**Задание:** написать класс матрица

**Код задания:**

#include "matrix.h"

#include <random>

#include <sstream>

Matrix::Matrix(int num\_rows, int num\_cols, Generator\* generator)

: data({})

{

if (num\_rows <= 0 || num\_cols <= 0 || generator==nullptr)

{

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

}

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

{

std::vector<int> temp;

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

{

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

}

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

}

}

void Matrix::add\_row(std::vector<int> user\_data)

{

if (user\_data.size() != this->data.size())

{

std::out\_of\_range("Invalide size new data");

}

this->data.push\_back(user\_data);

}

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

{

if (index>=rows\_counts())

{

throw std::out\_of\_range("Incorect index");

}

return data[index];

}

size\_t Matrix::rows\_counts() const

{

return this->data.size();

}

size\_t Matrix::cols\_counts() const

{

return this->data.size();

}

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

{

std::stringstream buffer;

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

{

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

{

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

}

buffer << std::endl;

}

return buffer.str();

}

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

{

return os << matrix.toString();

}

#pragma once

#include <vector>

#include <iostream>

#include <ostream>

#include <sstream>

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

class Matrix

{

public:

/\*\*

\*@brief Инициализация объекта типа матрицы по параметрам пользователя

\*@param num\_rows Количество строк

\*@param num\_cols Количество столбцов

\*@param generator Генератор, которым будет заполнена матрица

\*/

Matrix(int num\_rows, int num\_cols, Generator\* generator);

/\*\*

\*@brief Добавляется строка в матрицу по индексу, если же индекс не указан, то элемент добавляется в конец

\*@param user\_data Строка для добавления

\*/

void add\_row(std::vector<int> user\_data);

/\*\*

\*@brief Получение кол-ва строк в матрице

\*@return Кол-во строк в матрице

\*/

size\_t rows\_counts() const;

/\*\*

\*@brief Получение кол-ва столбцов в матрице

\*@return Кол-во столбцов в матрице

\*/

size\_t cols\_counts() const;

/\*\*

\*@brief Преобразование объекта матрицы в строку

\*@return Строка

\*/

std::string toString() const noexcept;

/\*\*

\*@brief Перегрузка оператора вывода

\*@param os - поток вывода

\*@param matrix - матрица, данные которые буду выведены

\*@return Поток вывода информации

\*/

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

/\*\*

\*@brief Перегрузка оператора квадратных скобок

\*@param index - индекс элемента в массиве

\*@return Элемент из матрицы стоящий под нужным индексом

\*/

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

private:

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

};

#include "Task.h"

Task::Task(Matrix matrix, Generator\* generator):

matrix(matrix),generator(generator)

{

}

Matrix Task::task1()

{

for (size\_t i = 0; i < 3; i++) {

for (size\_t j = 0; j < matrix.rows\_counts(); j++) {

matrix[j][i] = matrix[j][i] \* matrix[j][i];

}

}

return matrix;

}

﻿#pragma once

/\*\*

\*@brief Класс генератор

\*/

class Generator

{

public:

/\*\*

\*@brief Виртуальный метод для генерации числа

\*/

virtual int generate() = 0;

/\*\*

\*@brief Виртуальный деструктор,кот освобождает динамическую память объекта класса

\*/

virtual ~Generator() = 0 {};

};

#pragma once

#include "Generator.h"

#include <iostream>

class IStreamGenerator : public Generator

{

private:

std::istream& in;

public:

IStreamGenerator(std::istream & = std::cin);

int generate() override;

};

#include "IStreamGenerator.h"

IStreamGenerator::IStreamGenerator(std::istream& in)

:in{ in }

{

}

int IStreamGenerator::generate()

{

int value = 0;

this->in >> value;

return value;

}

﻿#pragma once

#include "Generator.h"

#include <random>

/\*\*

\*@brief Класс генератор случайного числа в заданном диапазоне

\*/

class RandomGenerator : public Generator

{

private:

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

std::mt19937 generator;

public:

/\*\*

\*@brief Инициализация объекта типа рандомгенератор

\*@param min - Минимальное значение

\*@param max - Максимальное значение

\*/

RandomGenerator(const int min, const int max);

/\*\*

\*@brief Генерирует какое-то число из заданного выше диапазона

\*@return Рандомные элементы из диапазона

\*/

int generate() override;

};

#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);

}

**Тесты:**

#include "pch.h"

#include "CppUnitTest.h"

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

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

#include"../Matrixx/Task.h"

#include"../Matrixx/matrix.h"

#include <iostream>

#include<vector>

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace TestMatrix

{

TEST\_CLASS(TestMatrix)

{

public:

TEST\_METHOD(MatrixAddRow\_Success)

{

RandomGenerator rg(1, 10);

Matrix m1(4, 4, &rg);

size\_t rows = m1.rows\_counts();

std::vector <int> vec = { 1, 2, 3, 0 };

m1.add\_row(vec);

size\_t new\_rows = m1.rows\_counts();

bool are\_equal = rows == new\_rows;

Assert::IsFalse(are\_equal);

}

TEST\_METHOD(Constructor\_ValitData\_Success)

{

RandomGenerator rg(1, 10);

Matrix m1(4, 4, &rg);

Assert::IsNotNull(&m1);

}

TEST\_METHOD(MatrixToString\_ValitData\_Success)

{

std::istringstream in("1 2 3 4");

IStreamGenerator isg(in);

Matrix m1(2, 2, &isg);

bool are\_equal = (m1.toString() == "1 2 \n3 4 \n");

Assert::IsTrue(are\_equal);

}

TEST\_METHOD(GetCols\_ValitData\_Success)

{

RandomGenerator rg(1, 10);

Matrix m1(4, 4, &rg);

bool are\_equal = (m1.cols\_counts() == 4);

Assert::IsTrue(are\_equal);

}

TEST\_METHOD(GetRows\_ValitData\_Success)

{

RandomGenerator rg(1, 10);

Matrix m1(2, 3, &rg);

bool are\_equal = (m1.rows\_counts() == 2);

Assert::IsTrue(are\_equal);

}

TEST\_METHOD(MatrixIndex\_ValitData\_Success)

{

std::istringstream in("1 2 3 4");

IStreamGenerator isg(in);

Matrix m1(2, 2, &isg);

bool are\_equal = (m1[0][1] == 2);

Assert::IsTrue(are\_equal);

}

TEST\_METHOD(Task\_ValitData\_Success)

{

std::istringstream in("1 2 3");

IStreamGenerator isg(in);

Matrix m1(1, 3, &isg);

Task t1(m1, &isg);

Matrix m2 = t1.task1();

bool are\_equal = (m2.toString() == "1 4 9 \n");

Assert::IsTrue(are\_equal);

}

};

**UML Диаграмма:**

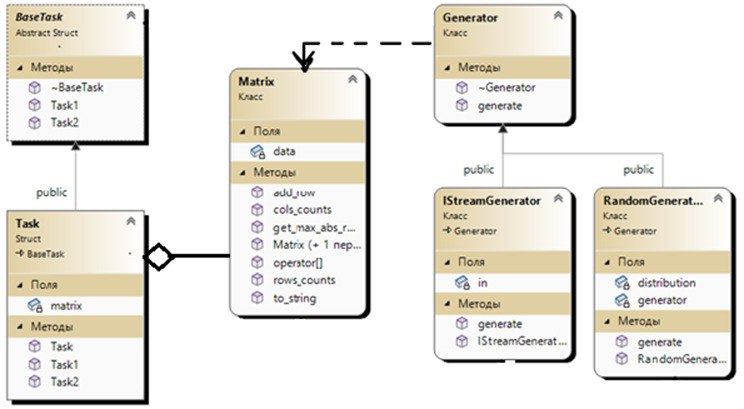
****

Рисунок 1 – UML диаграмма

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 2 – Approve