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

Институт транспортной техники и систем управления
Кафедра «Управление и защита информации»

Отчёт
по практической работе №4
По дисциплине «Языки программирования»

Выполнил: студент группы ТКИ — 241 ИТТСУ
Сапожников С. М.
Вариант №22
Проверила: Доцент
Васильева М. А.

Москва 2023

Формулировка задания:

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.

Код программ:

class one_dimension_arr

```
#pragma once
#include <cstddef>


class one_dimension_arr
{
   private:
   size_t num_el;
   int *my_arr;

   public:

one_dimension_arr(int num_el);

one_dimension_arr();

/**
 * @brief Уничтожает массив
```

```
 *
 */
~one_dimension_arr();

/**
 * @brief Construct a new one d arr::one d arr object
 *
 * @param arr
 */
one_dimension_arr(const one_dimension_arr &arr);

/**
 * @brief Construct a new one d arr::one d arr object
 *
 * @param other
 */
one_dimension_arr(one_dimension_arr&& other) noexcept;

/**
 * @brief
 *
 * @param other
 * @return one_d_arr&
 */
one_dimension_arr operator=(const one_dimension_arr &other);

/**
 * @brief
 *
 * @param other
 * @return one_d_arr&
 */
one_dimension_arr operator=(one_dimension_arr &&other) noexcept;

int operator[](int index);


/**
 * @brief
 *
 * @param lha
 * @param rha
 */
void swap(one_dimension_arr& lha, one_dimension_arr& rha);
```

```
size_t get_num_el();

int* get_arr();

};


#include "one_dimension_arr.h"
#include <iostream>
#include <stdexcept>
#include <utility>


one_dimension_arr::one_dimension_arr(int num_el)
{
   if (num_el <= 0)
   {
      throw std::logic_error("Размер должен быть положительным");
   }
   num_el = static_cast<size_t>(num_el);
   my_arr = new int[num_el];
   this->num_el = num_el;
}

one_dimension_arr::one_dimension_arr()
{
   this->num_el = 0;
   this->my_arr = nullptr;
}

one_dimension_arr::~one_dimension_arr()
{
   delete[] my_arr;
}


one_dimension_arr::one_dimension_arr(const one_dimension_arr& arr):
one_dimension_arr(arr.num_el)
{
   this->num_el = arr.num_el;
   this->my_arr = new int[num_el];
   for (size_t i = 0; i < num_el; ++i)
   {
         this->my_arr[i] = arr.my_arr[num_el];
   }
```

```
}

one_dimension_arr one_dimension_arr::operator=(const one_dimension_arr
&other)
{
    if (this == &other)
    {
        return *this;
    }
    one_dimension_arr n_arr(other);
    swap(*this, n_arr);
    return *this;
}

one_dimension_arr::one_dimension_arr(one_dimension_arr &&other) noexcept
{
    swap(*this, other);
}
one_dimension_arr one_dimension_arr::operator=(one_dimension_arr
&&other) noexcept
{
    if (this == &other)
    {
        return *this;
    }
    swap(*this, other);
    return *this;
}

int one_dimension_arr::operator[](int index)
{
    return my_arr[index];
}

size_t one_dimension_arr::get_num_el()
{
    return num_el;
}

int *one_dimension_arr::get_arr()
{
    return my_arr;
}
```

```
void one_dimension_arr::swap(one_dimension_arr& lha, one_dimension_arr&
rha)
{
    std::swap(lha.my_arr, rha.my_arr);
    std::exchange(lha.num_el, rha.num_el);
}
```

class two_dimension_arr

```
#include "one_dimension_arr.h"
#include "one_dimension_arr.cpp"
#include "initializer_list"
#pragma once

class two_dimension_arr
{
    friend class one_dimension_arr;
    private:
    size_t num_rows;
    size_t num_columns;
    int **my_matrix;

    public:

/**
 * @brief Construct a new two d arr object
 *
 * @param num_rows
 * @param num_columns
 */
two_dimension_arr(int num_rows, int num_columns);

/**
 * @brief Destroy the two d arr object
 *
 */
~two_dimension_arr();

two_dimension_arr(size_t num_rows, size_t num_columns,
std::initializer_list<int> data);

/**
 * @brief Construct a new Array object
 *
 * @param arr
```

```
 */
two_dimension_arr(const two_dimension_arr& matrix);

size_t get_num_rows();

size_t get_num_columns();


/**
 * @brief
 *
 */
two_dimension_arr(two_dimension_arr&& other) noexcept;

two_dimension_arr operator=(const two_dimension_arr &other);

two_dimension_arr operator=(two_dimension_arr &&other) noexcept;

friend void swap(two_dimension_arr& lha, two_dimension_arr& rha);

int* operator[](int num_rows);

bool operator==(two_dimension_arr& other) noexcept;

friend std::ostream& operator<<(std::ostream& os, two_dimension_arr&
my_matrix);
};

#include "two_dimension_arr.h"
#include "one_dimension_arr.h"
#include <cstddef>
#include <stdexcept>
#include<utility>
#include <sstream>
#include <iostream>
#include <initializer_list>
#include <string>


two_dimension_arr::two_dimension_arr(int num_rows, int num_columns)
{
    if (num_rows <= 0 || num_columns <= 0)
    {
        throw std::logic_error("Размер должен быть положительным");
    }
```

```
    num_rows = static_cast<size_t>(num_rows);
    num_columns = static_cast<size_t>(num_columns);
    this->num_rows = num_rows;
    this->num_columns = num_columns;
    my_matrix = new int*[num_rows];
    for (size_t i = 0; i < num_rows; i++)
    {
        my_matrix[i] = one_dimension_arr(num_columns).get_arr();
    }
}

two_dimension_arr::two_dimension_arr(size_t num_rows, size_t num_columns,
std::initializer_list<int> data)
        :my_matrix(new int*[num_rows])
{
        size_t i = 0;
        size_t j = 0;
        for (auto& item : data)
                {
                my_matrix[i][j] = item;
                i++;
                if (i == num_columns)
                {
                        j++;
          i = 0;
                }
        }

}

two_dimension_arr::~two_dimension_arr()
{
    for (auto i = 0; i < num_rows; i++)
    {
        delete[] my_matrix[i];
    }
    delete[] this->my_matrix;
}

two_dimension_arr::two_dimension_arr(const two_dimension_arr &matrix)
{
    this->num_rows = matrix.num_rows;
    this->num_columns = matrix.num_columns;
    this->my_matrix = new int*[num_rows];
    for (size_t i = 0; i < num_rows; ++i)
```

```
    {
        this->my_matrix[i] = one_dimension_arr(num_rows).get_arr();
        for (int j = 0; j < num_columns; ++j)
        {
            this->my_matrix[i][j] = matrix.my_matrix[i][j];
        }
    }
}

size_t two_dimension_arr::get_num_rows()
{
    return num_rows;
}

size_t two_dimension_arr::get_num_columns()
{
    num_columns = static_cast<int>(num_columns);
    one_dimension_arr arr = one_dimension_arr(num_columns);
    return arr.get_num_el();
}



two_dimension_arr::two_dimension_arr(two_dimension_arr &&other) noexcept
{
    swap(*this,other);
}

two_dimension_arr two_dimension_arr::operator=(const two_dimension_arr
&other)
{
        if (this == &other)
    {
        return *this;
    }
    two_dimension_arr n_arr(other);
    swap(*this, n_arr);
    return *this;
}

two_dimension_arr two_dimension_arr::operator=(two_dimension_arr
&&other) noexcept
{
    if (this == &other)
    {
        return *this;
```

```
    }
    std::swap(this->num_columns, other.num_columns);
    std::swap(this->num_rows, other.num_rows);
    std::swap(this->my_matrix, other.my_matrix);
    return *this;
}

void swap(two_dimension_arr &lha, two_dimension_arr &rha)
{
    for (size_t i = 0; i < lha.num_rows; i++)
    {
        std::swap(lha.my_matrix[i], rha.my_matrix[i]);
    }
    std::swap(lha.num_columns, rha.num_columns);
    std::swap(lha.num_rows, rha.num_rows);
    std::swap(lha.my_matrix, rha.my_matrix);
}

int* two_dimension_arr::operator[](int num_rows)
{
    return my_matrix[num_rows];
}

std::ostream& operator<<(std::ostream &os, two_dimension_arr &my_matrix)
{
    size_t num_rows = my_matrix.get_num_rows();
    size_t num_columns = my_matrix.get_num_columns();
    std::stringstream result_string;
    for (int i = 0; i < num_rows; i++)
    {
        for (int j = 0; j < num_columns; j++)
        {
            result_string << my_matrix[i][j] << " ";
        }
        result_string << "\n";
    }
    return os << result_string.str();
}

bool two_dimension_arr::operator==(two_dimension_arr& other) noexcept
{
    for (int i = 0; i < num_rows; i++)
    {
        for (int j = 0; j < num_columns; j++)
        {
```

```
            if ( this->my_matrix[i][j] != other.my_matrix[i][j] )
            return false;
        }


    }
return true;
}
```

class Generator

```
#pragma once
class Generator
{
    public:
    virtual ~Generator() = 0;
    virtual int generate() = 0;
};
```

```
#include "Generator.h"
#pragma once


int Generator::generate(){ return 0;}

Generator::~Generator() { }
```

class ManualGenerator

```
#include "Generator.h"
#include "Generator.cpp"
#include <iostream>
#pragma once

class ManualGenerator: public Generator
{
    private:
    std::istream& in;
    public:



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

int generate() override;
};
```

```
#include <iostream>
#include "ManualGenerator.h"
#pragma once
ManualGenerator::ManualGenerator(std::istream &in): in{in}{}

int ManualGenerator::generate()
{
    int value = 0;
    this->in >> value;
    return value;
}
```

class RandomGenerator

```
#include "Generator.h"
#include "Generator.cpp"
#include <random>
#pragma once

class RandomGenerator: public Generator
{
    private:
    std::uniform_int_distribution<int> distribution;
    std::mt19937 generator;
    public:

RandomGenerator(const int min, const int max);

int generate() override;
};

#include "RandomGenerator.h"
#include "Generator.h"
#include "Generator.cpp"
#include <random>
#pragma once

int RandomGenerator::generate()
{
    return this->distribution(this->generator);
}

RandomGenerator::RandomGenerator(const int min, const int max)
{
    this->generator = std::mt19937(std::random_device{}());
    this->distribution = std::uniform_int_distribution<int>(min, max);
```

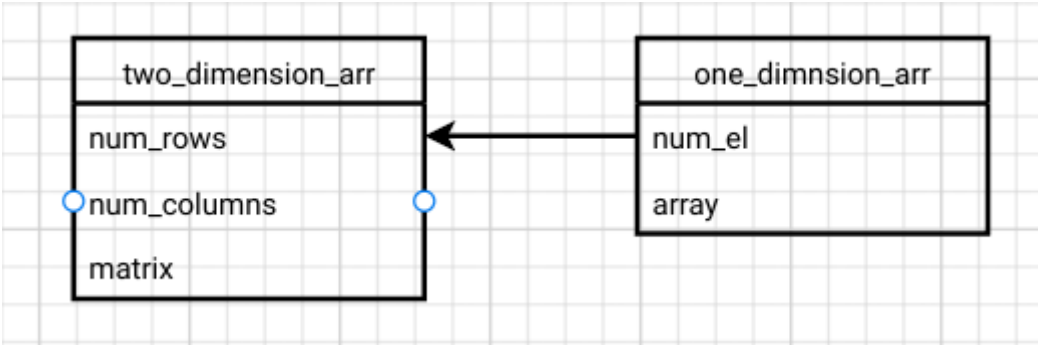
```
}
```

class Exercise

```
#include "two_dimension_arr.h"

class Exercise
{
   public:

   virtual void task_1();

   virtual int** task_2();
};
```

```
#include "Exercise.h"

void Exercise::task_1(){}

int** Exercise::task_2(){}
```

UML-диаграмма



выполненные тесты:

```
[sergey@sergey-bohkwax9x ThirdSemester]$ g++ test.cpp -lgtest
[sergey@sergey-bohkwax9x ThirdSemester]$ ./a.out
[==========] Running 6 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 6 tests from MatrixMetodsTest
[ RUN      ] MatrixMetodsTest.GetRowsTest
[       OK ] MatrixMetodsTest.GetRowsTest (0 ms)
[ RUN      ] MatrixMetodsTest.GetColumnsTest
[       OK ] MatrixMetodsTest.GetColumnsTest (0 ms)
[ RUN      ] MatrixMetodsTest.SwapTest
[       OK ] MatrixMetodsTest.SwapTest (0 ms)
[ RUN      ] MatrixMetodsTest.ReplaceMaxElTest
[       OK ] MatrixMetodsTest.ReplaceMaxElTest (0 ms)
[ RUN      ] MatrixMetodsTest.ArrayInsert
[       OK ] MatrixMetodsTest.ArrayInsert (0 ms)
[ RUN      ] MatrixMetodsTest.PrintTest
[       OK ] MatrixMetodsTest.PrintTest (0 ms)
[----------] 6 tests from MatrixMetodsTest (0 ms total)

[----------] Global test environment tear-down
[==========] 6 tests from 1 test suite ran. (0 ms total)
[  PASSED  ] 6 tests.
```

Полученный аппрув:

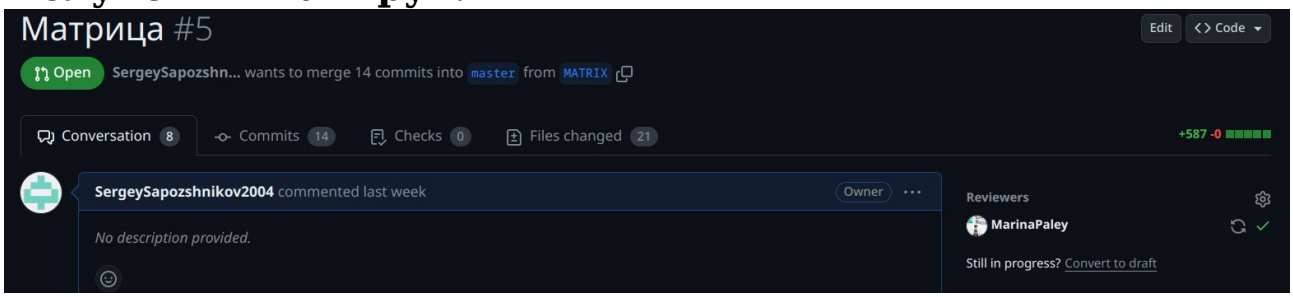


Рисунок 1- аппрув