Министерство образования Республики Беларусь
Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра электронных вычислительных машин

ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ № 1
Классы и объекты

по дисциплине «Программирование на языках высокого уровня»

Выполнил ст. гр. 450503 А.П. Красько

Проверил асс. каф. ЭВМ И.Г. Скиба

Минск 2025

1 ПОСТАНОВКА ЗАДАЧИ

Создать класс матрица. Память под матрицу выделять динамически. Определить конструктор без параметров, конструктор с параметрами. Реализовать методы: ввод данных в матрицу, вывод матрицы на экран, вычитание числа из элементов матрицы. Проверить работу методов этого класса.

2 ЛИСТИНГ КОДА

Файл main.cc

```
#include <functional>

#include "../../lib/consoleUtils.hh"
#include "matrix.hh"
#include "screens.hh"

using namespace std;
using namespace console_utils;
using namespace screen_handlers;

int main(void) {
    mat::Matrix matrix;
    array<function<bool(mat::Matrix&)>, 4> actions = {inputMatrix, printMatrix,
subtractFromElement,
                                                      [](const mat::Matrix&) {
return 0; }};
    unsigned int response;
    do {
        printMainScreen();
        readT(response, ">", [](unsigned int num) { return num > 0 && num <= 4;
});
        cout << "\x{1B}[2J\x{1B}[H\n";
    } while (actions[response - 1](matrix));
    return 0;
}
```

Файл matrix.hh

```
#pragma once
#include <memory>

namespace mat {
using initializer_list = std::initializer_list<double>;
using initializer_matrix = std::initializer_list<initializer_list>;
class Matrix {
   private:
    size_t rows_;
    size_t cols_;
    double *data_;

   private:
    void copyFrom(const initializer_matrix &mat);
```

```
   public:
    Matrix();
    Matrix(size_t rows, size_t cols);
    explicit Matrix(Matrix &&other) noexcept;
    explicit Matrix(const Matrix &other);
    Matrix(const initializer_matrix &mat);
    ~Matrix();

    const Matrix &insert(const initializer_matrix &mat);
    Matrix &resize(size_t rows, size_t cols);

    size_t getRows() const;
    size_t getCols() const;
    double &getElement(size_t i, size_t j) const;
    const Matrix &setElement(size_t i, size_t j, double el) const;
    int isElem(size_t i, size_t j) const;
    void print() const;
    const Matrix &subtractFromElement(size_t i, size_t j, double val) const;

    Matrix &operator=(const initializer_matrix &mat);
    Matrix &operator=(const Matrix &other);
    Matrix &operator=(Matrix &&other) noexcept;
};

}  // namespace mat
```

Файл matrix.cc

```
#include "matrix.hh"

#include <iostream>
#include <print>
#include <ranges>
#include <stdexcept>

using namespace mat;

Matrix::Matrix(size_t rows, size_t cols) : rows_{rows}, cols_{cols},
data_(new double[cols_ * rows_]){};

Matrix::Matrix() : Matrix(2, 2){};

Matrix::Matrix(Matrix &&other) noexcept :  rows_{other.rows_},
cols_{other.cols_}, data_{other.data_} {};

Matrix::Matrix(const Matrix &other) : Matrix(other.cols_, other.rows_) {
    std::copy(other.data_, other.data_ + cols_ * rows_, data_);

}

Matrix::Matrix(const initializer_matrix &mat) : Matrix(mat.size(),
mat.begin()->size()) { copyFrom(mat); };

Matrix::~Matrix() { delete[] data_; }

void Matrix::copyFrom(const initializer_matrix &mat) {
    for (size_t i = 0; i < rows_; i++) {
        if ((mat.begin() + i)->size() != cols_) throw
std::invalid_argument("Invalid initializer list");
        const initializer_list line = *(mat.begin() + i);
        std::ranges::copy(line, data_ + i * cols_);
    }
```

```
}

const Matrix &Matrix::insert(const initializer_matrix &mat) {
    resize(mat.size(), mat.begin()->size());
    copyFrom(mat);
    return *this;
}

Matrix &Matrix::resize(size_t rows, size_t cols) {
    size_t oldLen = rows_ * cols_;
    rows_ = rows;
    cols_ = cols;
    if (oldLen < rows_ * cols_) {
        delete[] data_;
        data_ = new double[rows_ * cols_];
    }
    return *this;
}

size_t Matrix::getRows() const { return rows_; }

size_t Matrix::getCols() const { return cols_; }

void Matrix::print() const {
    size_t maxWidth = 0;
    for (int i = 0; i < rows_; i++) {
        for (int j = 0; j < cols_; j++) {
            maxWidth = std::max(maxWidth, std::format("{}g", data_[i * cols_
+ j]).size());
        }
    }
    maxWidth--;
    for (size_t i = 0; i < rows_; i++) {
        std::cout << '|';
        for (size_t j = 0; j < cols_ - 1; j++) {
            std::print("{:<{}} ", data_[i * cols_ + j], maxWidth);
        }
        std::print("{:<{}}|\n", data_[i * cols_ + cols_ - 1], maxWidth);
    }
}

int Matrix::isElem(size_t i, size_t j) const { return i < rows_ && j < cols_;
}

double &Matrix::getElement(size_t i, size_t j) const {
    if (!isElem(i, j)) throw std::invalid_argument("Indexes out of range");
    return data_[i * cols_ + j];
}

const Matrix &Matrix::setElement(size_t i, size_t j, double el) const{
    getElement(i, j) = el;
    return *this;
}

const Matrix &Matrix::subtractFromElement(size_t i, size_t j, double val)
const{
    getElement(i, j) -= val;
    return *this;
}

Matrix &Matrix::operator=(const initializer_matrix &mat) {
    copyFrom(mat);
    return *this;
```

```
}

Matrix &Matrix::operator=(const Matrix &other) {
    resize(other.rows_, other.cols_);
    std::ranges::copy(other.data_, other.data_ + cols_ * rows_, data_);
    return *this;
}

Matrix &Matrix::operator=(Matrix &&other) noexcept {
    delete[] data_;
    cols_ = other.cols_;
    rows_ = other.rows_;
    data_ = other.data_;
    return *this;
}
```

Файл screens.cc

```
#include <iostream>
#include <print>
#include "matrix.hh"
#include "../../lib/consoleUtils.hh"

using namespace console_utils;
using namespace std;
namespace screen_handlers {
bool inputMatrix(mat::Matrix& mat) {
    size_t rows;
    size_t cols;
    auto check = [](size_t num) { return num > 0; };
    readT(rows, "Please enter number of matrix rows: ", check);
    readT(cols, "Please enter number of matrix collumns: ", check);

    mat.resize(rows, cols);
    std::cout << rows << " " << cols;
    for (size_t i = 0; i < rows; i++) {
        for (size_t j = 0; j < cols; j++) {
            double element;
            readT(element, std::format("Please enter element with index
[{}][{}]:",i + 1, j+1 ));
            mat.setElement(i, j, element);
        }
    }
    return true;
}
bool printMatrix(const mat::Matrix& mat) { mat.print(); return true;}

bool subtractFromElement(const mat::Matrix& mat) {
    size_t row;
    size_t col;
    double op;
    readT(
        row, "Please enter elemet row: ", [&mat](const size_t& num) { return
num > 0 && num <= mat.getRows(); },
        "Invalid value\n");
    readT(
        col, "Please enter elemet collumn: ", [&mat](const size_t& num) {
return num > 0 && num <= mat.getCols(); },
        "Invalid value\n");
    readT(op, "Please enter number to subtruct: ");
    mat.subtractFromElement(row - 1, col - 1, op);
    return true;
```

```
}
void printMainScreen() {
    auto [cols, rows] = getConsoleDimensions();
    println("{:^{}}", "\x{1B}[48;5;35mLab 1\x{1B}[0m", cols);
    println("Please select action:\n");
    println("    1.Input matrix");
    println("    2.Print matrix");
    println("    3.Subtruct from matrix element");
    println("    4.Exit");
}
}  // namespace screen_handlers
```

Файл screens.hh

```
#pragma once

#include "matrix.hh"

namespace screen_handlers {
bool inputMatrix(mat::Matrix& mat);
bool printMatrix(const mat::Matrix& mat);
bool subtractFromElement(const mat::Matrix& mat);
void printMainScreen();
}  // namespace screen_handlers
```

Файл consoleUtils.cc

```
#include <iostream>

#ifdef __linux__
#include <sys/ioctl.h>
#include <unistd.h>
#endif

#ifdef _WIN32
#include <windows.h>
#endif

namespace console_utils {

std::pair<int, int> getConsoleDimensions() {
#ifdef _WIN32
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE), &csbi);
    return std::make_pair(csbi.srWindow.Right - csbi.srWindow.Left + 1,
csbi.srWindow.Bottom - csbi.srWindow.Top + 1);
#endif
#ifdef __linux__
    struct winsize w;
    ioctl(STDOUT_FILENO, TIOCGWINSZ, &w);
    return std::make_pair(w.ws_col, w.ws_row);
#endif
}
}  // namespace console_utils
```

Файл consoleUtils.hh

```
#pragma once
#include <iostream>
```

```
#include <limits>

namespace console_utils {


std::pair<int, int> getConsoleDimensions();


template <typename T, typename CT>
void readT(T& data, const std::string& massage, CT bound) {
    std::cout << massage;
    while (((std::cin >> data).fail()) || !bound(data)) {
        std::cout << "Invalid input. Reread input requierments\n";
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), ' ');
        std::cout << massage;
    }
    std::cin.clear();
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}
template <typename T>
void readT(T& data, const std::string& massage) {
    std::cout << massage;
    while ((std::cin >> data).fail()) {
        std::cout << "Invalid input. Reread input requierments\n";
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), ' ');
        std::cout << massage;
    }
    std::cin.clear();
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}
template <typename T, typename CT>
void readT(T& data, const std::string& message, CT bound, const std::string&
errmess) {
    std::cout << message;
    while (((std::cin >> data).fail()) || !bound(data)) {
        std::cout << errmess;
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        std::cout << message;
    }
    std::cin.clear();
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}
}  // namespace console_utils
```

3 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

Рисунок 4.1 – Главное меню

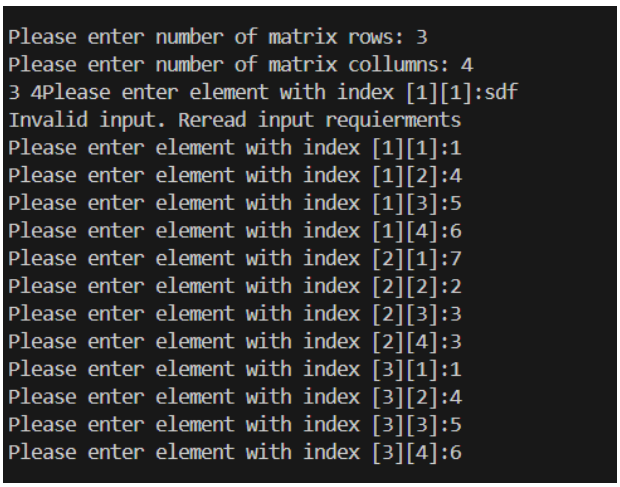

Рисунок 4.2 – Меню ввода

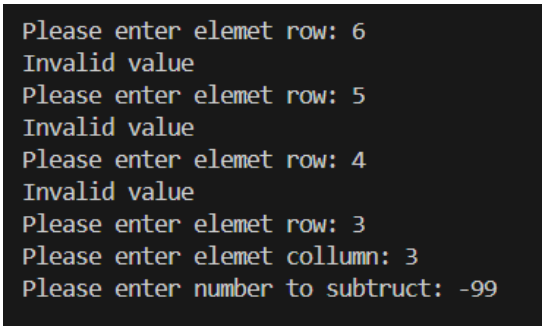

Рисунок 4.3 – Меню вычитания из элемента

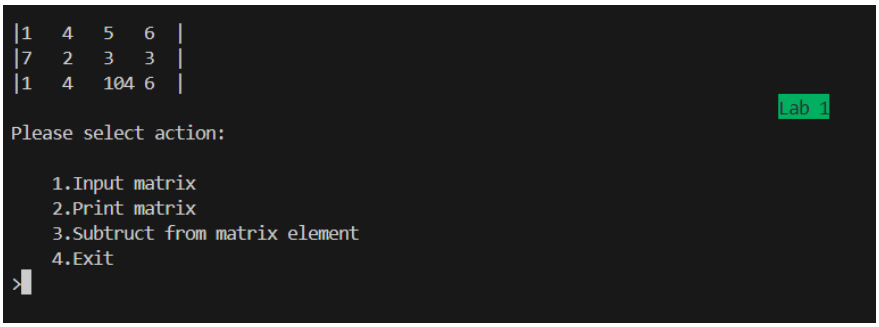

Рисунок 4.4 – Вывод матрицы

4 ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы был разработан класс matrix, реализующий основные операции для работы с матрицами: динамическое выделение памяти, ввод и вывод данных, вычитание числа из указанного элемента. Проверка методов класса подтвердила их корректную работу. Интерфейс программы обеспечивает удобное взаимодействие с пользователем через консольное меню.