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.cс

#include <functional>

#include <iostream>

#include <limits>

#include <print>

#include "matrix.hh"

#include "stringUtils.hh"

using namespace std;

using namespace str\_utils;

void 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);

cout << rows << " " << cols;

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

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

double element;

readT(element, format("Please enter element with index [{}][{}]:",i + 1, j+1 ));

mat.setElement(i, j, element);

}

}

}

void printMatrix(const mat::Matrix& mat) { mat.print(); }

void subtractFromElement(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);

}

string offest(size\_t offset) { return string(offset, ' '); }

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");

}

int main(void) {

mat::Matrix matrix;

array<function<void(mat::Matrix&)>, 4> actions = {inputMatrix, printMatrix, subtractFromElement,

[](const mat::Matrix&) { exit(0); }};

while (true) {

printMainScreen();

unsigned int response;

readT(response, ">", [](unsigned int num) { return num > 0 && num <= 4; });

cout << "\x{1B}[2J\x{1B}[H\n";

actions[response - 1](matrix);

}

}

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 <sstream>

#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 stringUtils.hh

#pragma once

#include <iostream>

#include <limits>

#ifdef \_\_linux\_\_

#include <sys/ioctl.h>

#include <unistd.h>

#endif

#ifdef \_WIN32

#include <windows.h>

#endif

namespace str\_utils {

#ifdef \_WIN32

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

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\_\_

#include <sys/ioctl.h>

#include <unistd.h>

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

struct winsize w;

ioctl(STDOUT\_FILENO, TIOCGWINSZ, &w);

return std::make\_pair(w.ws\_col, w.ws\_row);

}

#endif

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::cout << massage;

}

}

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::cout << massage;

}

}

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;

}

}

} // namespace str\_utils

**3 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**



Рисунок 4.1 – Главное меню

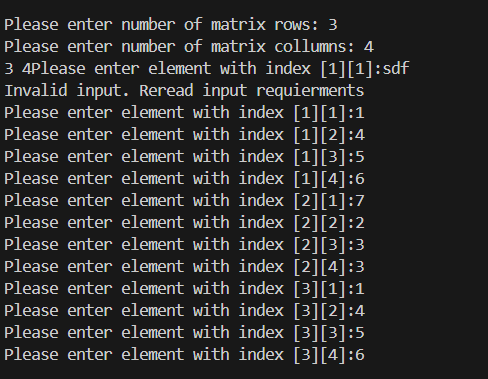


Рисунок 4.2 – Меню ввода

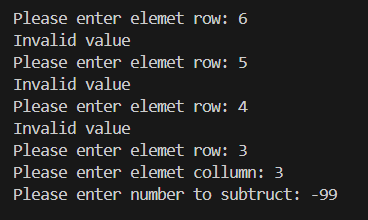


Рисунок 4.3 – Меню вычитания из элемента

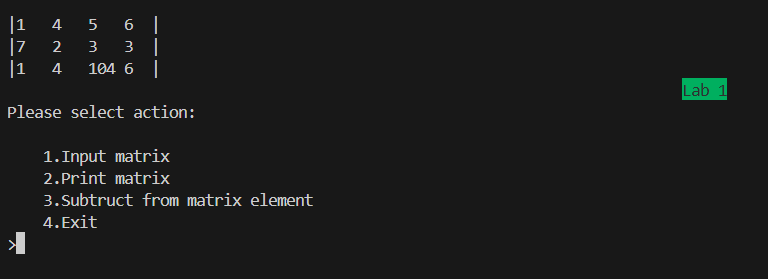


Рисунок 4.4 – Вывод матрицы

**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.