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 Д.С. Абрамчук

Проверил асс. каф. ЭВМ И.Г. Скиба

Минск 2025

# **1 ПОСТАНОВКА ЗАДАЧИ**

Создать класс, в котором реализовать функции для работы с матрицами:

а) функция производит  перемножение матриц;

б) функция производит сложение двух матриц.

Память под матрицы отводить динамически. Использовать конструктор с параметрами. Деструктор должен освобождать память, выделенную под матрицы.

# **2 ЛИСТИНГ КОДА**

Файл main.cpp

#include "function.h"  
  
int main() {  
 ShowMenu();  
 return 0;  
}

Файл function.cpp

#include "function.h"  
#include "iostream"  
  
enum TErrorCode {  
 *CORRECT*,  
 *NON\_NUMERIC*,  
 *OUT\_OF\_RANGE*};  
  
int MIN\_ARR = 2;  
int MAX\_ARR = 10;  
  
Matrix::Matrix(int row, int col) {  
 cols = col;  
 rows = row;  
 M = new float \*[rows];  
 for (int i = 0; i < rows; i++) {  
 M[i] = new float[cols];  
 }  
}  
  
Matrix::~Matrix() {  
 for (int i = 0; i < rows; i++) {  
 delete[] M[i];  
 }  
 delete[] M;  
}  
  
const std::string ERR[] = {  
 "",  
 "Error. Non-numeric value. Please try again. ",  
 "Error. Out of Range. Please try again. ",  
 "Error. incorrect size matrix. please try again."};  
  
void MatrixSetting(int &rows, int &cols) {  
 TErrorCode error;  
 int i;  
 do {  
 error = *CORRECT*;  
 std::cout << "Please write size of matrix through the space in the range " << MIN\_ARR << " .. " << MAX\_ARR  
 << std::endl;  
 std::cin >> rows;  
 std::cin >> cols;  
 if (std::cin.fail() || std::cin.peek() != '\n') {  
 error = *NON\_NUMERIC*;  
 std::cin.clear();  
 while (std::cin.get() != '\n');  
 }  
 if ((rows < MIN\_ARR || rows > MAX\_ARR) || (cols < MIN\_ARR || cols > MAX\_ARR)) {  
 error = *OUT\_OF\_RANGE*;  
 }  
 if (error != *CORRECT*) {  
 std::cout << ERR[error] << std::endl;  
 }  
 } while (error != *CORRECT*);  
}  
  
void Matrix::CreateMatrix() {  
 std::cout << "please write elements of the matrix through the space" << std::endl;  
 for (int i = 0; i < rows; i++) {  
 for (int j = 0; j < cols; j++) {  
 std::cin >> M[i][j];  
 }  
 }  
}  
  
void Matrix::ShowMatrix() {  
 for (int i = 0; i < rows; i++) {  
 for (int j = 0; j < cols; j++) {  
 std::cout << M[i][j] << " ";  
 }  
 std::cout << std::endl;  
 }  
}  
  
Matrix Matrix::ADD(const Matrix &B) const {  
 Matrix C(rows, cols);  
 for (int i = 0; i < rows; i++) {  
 for (int j = 0; j < cols; j++) {  
 C.M[i][j] = M[i][j] + B.M[i][j];  
 }  
 }  
 return C;  
}  
  
Matrix Matrix::MULTIPLY(const Matrix &B) const {  
 Matrix C(rows, B.cols);  
  
 for (int i = 0; i < rows; i++)  
 for (int j = 0; j < B.cols; j++) {  
 C.M[i][j] = 0;  
 for (int k = 0; k < cols; k++)  
 C.M[i][j] += M[i][k] \* B.M[k][j];  
 }  
  
 return C;  
}  
  
void ShowMenu() {  
 int choice;  
 do {  
 std::cout << "\n=== Menu ===\n";  
 std::cout << "1. Matrix addition\n";  
 std::cout << "2. Matrix multiplication\n";  
 std::cout << "0. exit\n";  
 std::cout << "choose point: ";  
 std::cin >> choice;  
  
 if (choice == 0) break;  
  
 int rows1, cols1;  
 std::cout << "Matrix A:\n";  
 MatrixSetting(rows1, cols1);  
 Matrix A(rows1, cols1);  
 A.CreateMatrix();  
  
 int rows2, cols2;  
 std::cout << "Matrix B:\n";  
 MatrixSetting(rows2, cols2);  
 Matrix B(rows2, cols2);  
 B.CreateMatrix();  
  
 if (choice == 1) {  
 if (rows1 == rows2 && cols1 == cols2) {  
 Matrix C = A.ADD(B);  
 std::cout << "Result Matrix:\n";  
 C.ShowMatrix();  
 } else {  
 std::cout << "ERROR, please try again.\n";  
 }  
 } else if (choice == 2) {  
 if (cols1 == rows2) {  
 Matrix D = A.MULTIPLY(B);  
 std::cout << "Result Matrix:\n";  
 D.ShowMatrix();  
 } else {  
 std::cout << "ERROR, please try again.\n";  
 }  
 } else {  
 std::cout << "ERROR, please try again.\n";  
 }  
  
 } while (true);  
  
 std::cout << "END\n";  
}

Файл function.h

#ifndef LAB\_\_\_1\_\_FUNCTION\_H  
#define LAB\_\_\_1\_\_FUNCTION\_H  
  
class Matrix {  
 int cols;  
  
 int rows;  
  
 float \*\*M;  
  
public:  
  
 Matrix(int row, int cols);  
  
 ~Matrix();  
  
 void CreateMatrix();  
  
 void ShowMatrix();  
  
 Matrix ADD(const Matrix &B) const;  
  
 Matrix MULTIPLY(const Matrix &B) const;  
  
};  
  
void MatrixSetting(int &rows, int &cols);  
  
void ShowMenu();  
  
#endif

**3 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

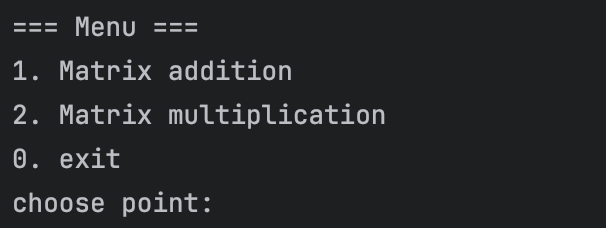
****

Рисунок 4.1 – Меню, выводимое на экран пользователя

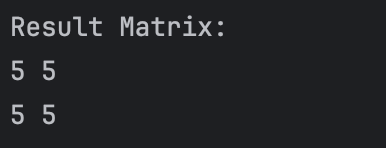


Рисунок 4.2 – Результат сложения матриц

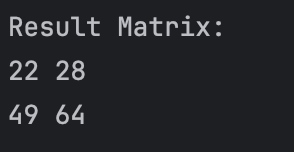


Рисунок 4.3 – Результат умножения матриц