**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра «АПУ»**

**ОТЧЁТ**

**по лабораторной работе № 4**

**по дисциплине «Программирование»**

**«ДВУМЕРНЫЕ И ДИНАМИЧЕСКИЕ МАССИВЫ»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 4391 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Зайцева О. Ю. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Писарев А.С. |

Санкт-Петербург

2024 г.

**Цель работы:**

Научиться работать с двумерными и динамическими массивами.

Написать программу, используя язык программирования C++.

**Ход работы:**

Была изучена задача на двумерные и динамические массивы.

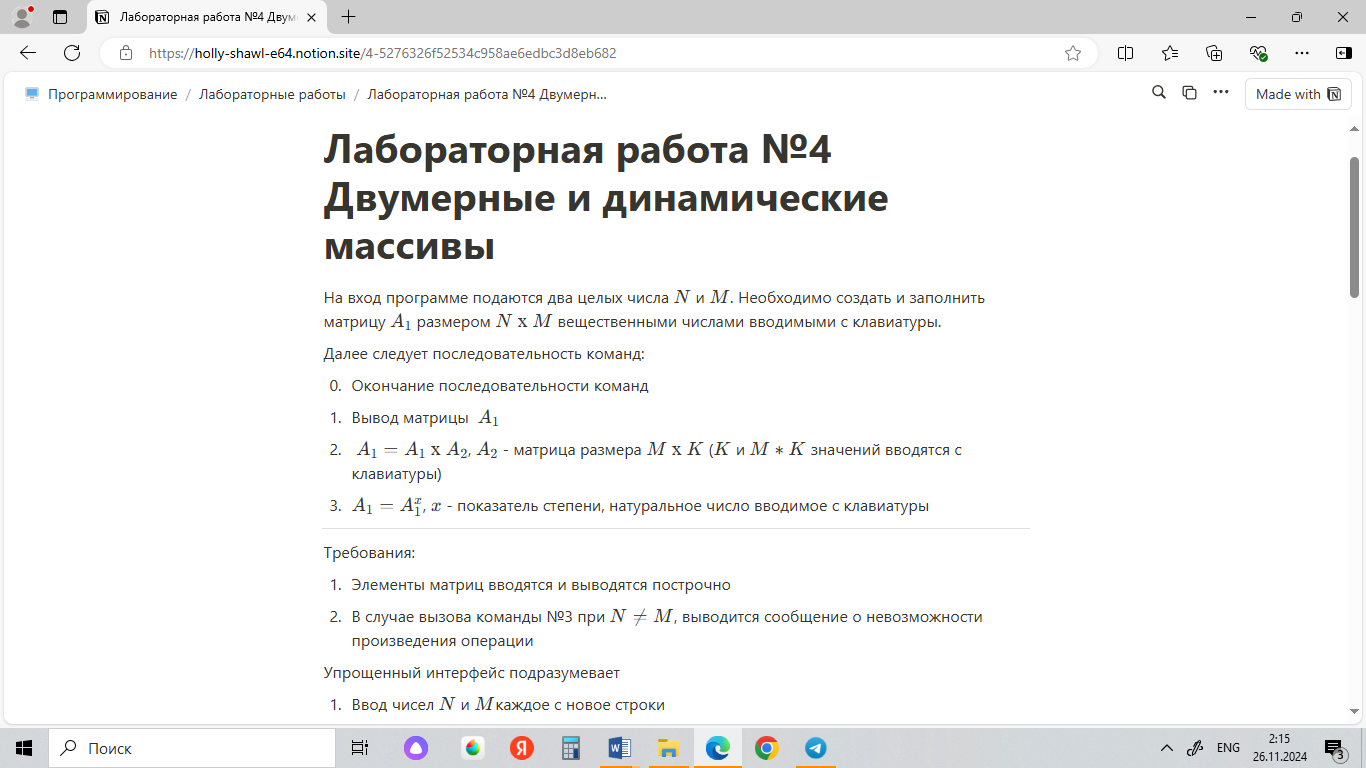


Рисунок 1 – Задача

В самом начале было выполнено подключение к библиотекам.

Библиотека <iostream> предназачена для организации ввода и вывода.

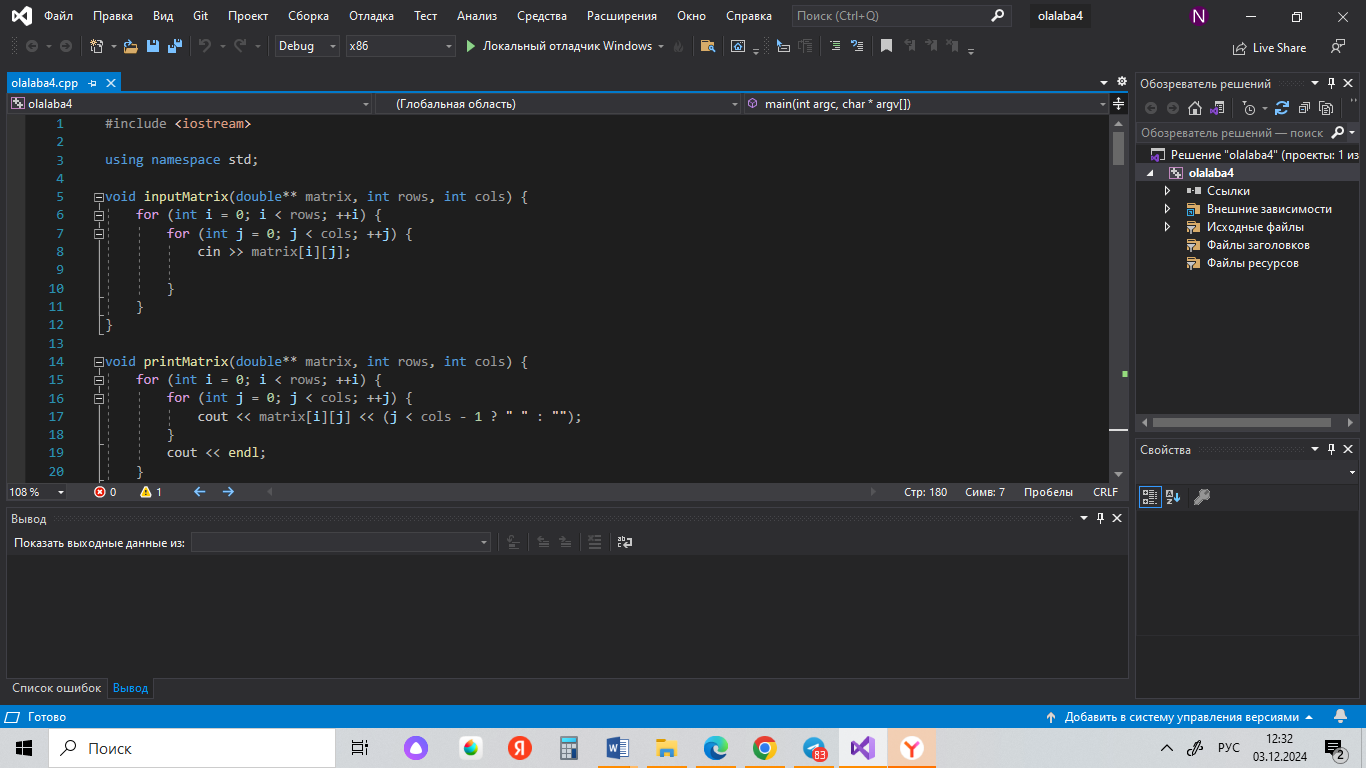


Рисунок 2 – Подключенная библиотека

Инструкция using namespace std представляет собой команду для использования стандартного пространства имён std.

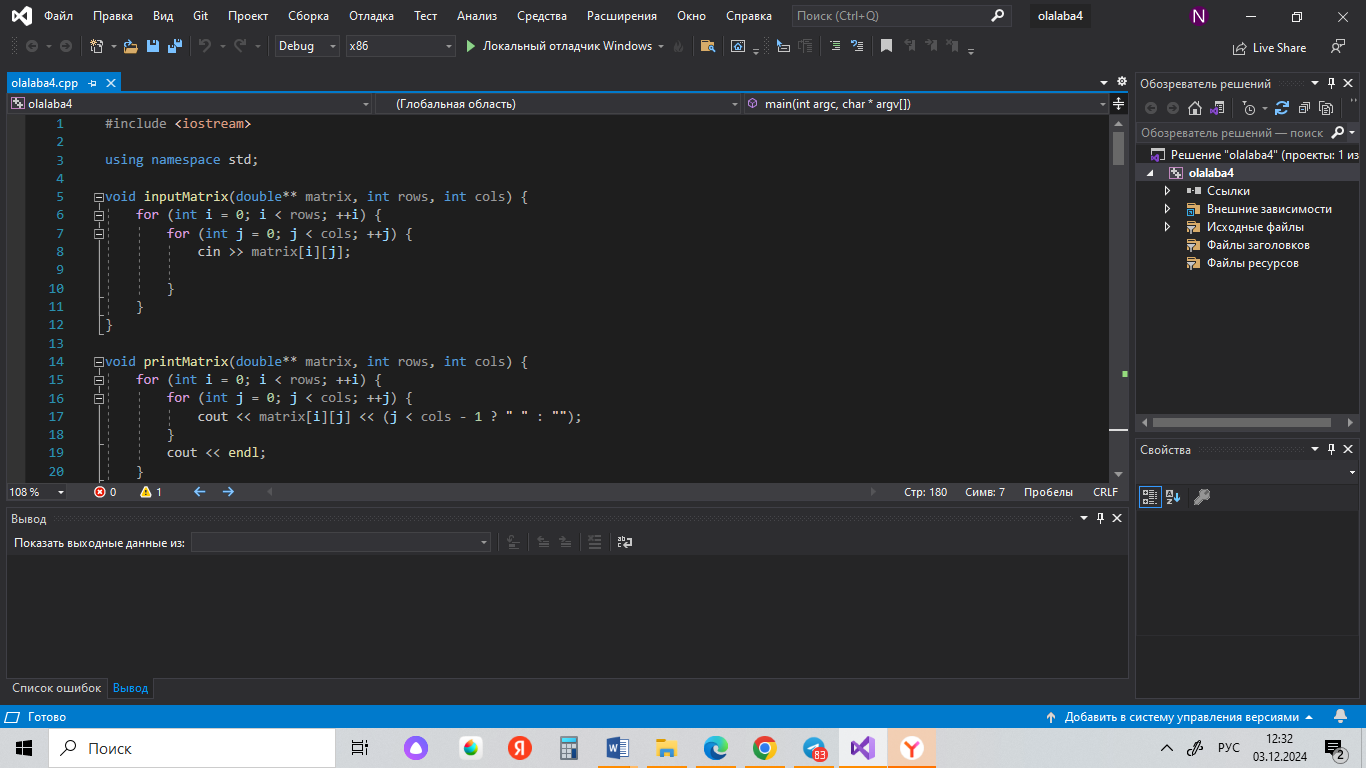


Рисунок 3 – Пространство имён

Была добавлена функция для ввода матрицы. Эта функция принимает указатель на матрицу (двумерный массив) и размеры матрицы (число строк и столбцов). Она заполняет матрицу значениями, введенными пользователем.

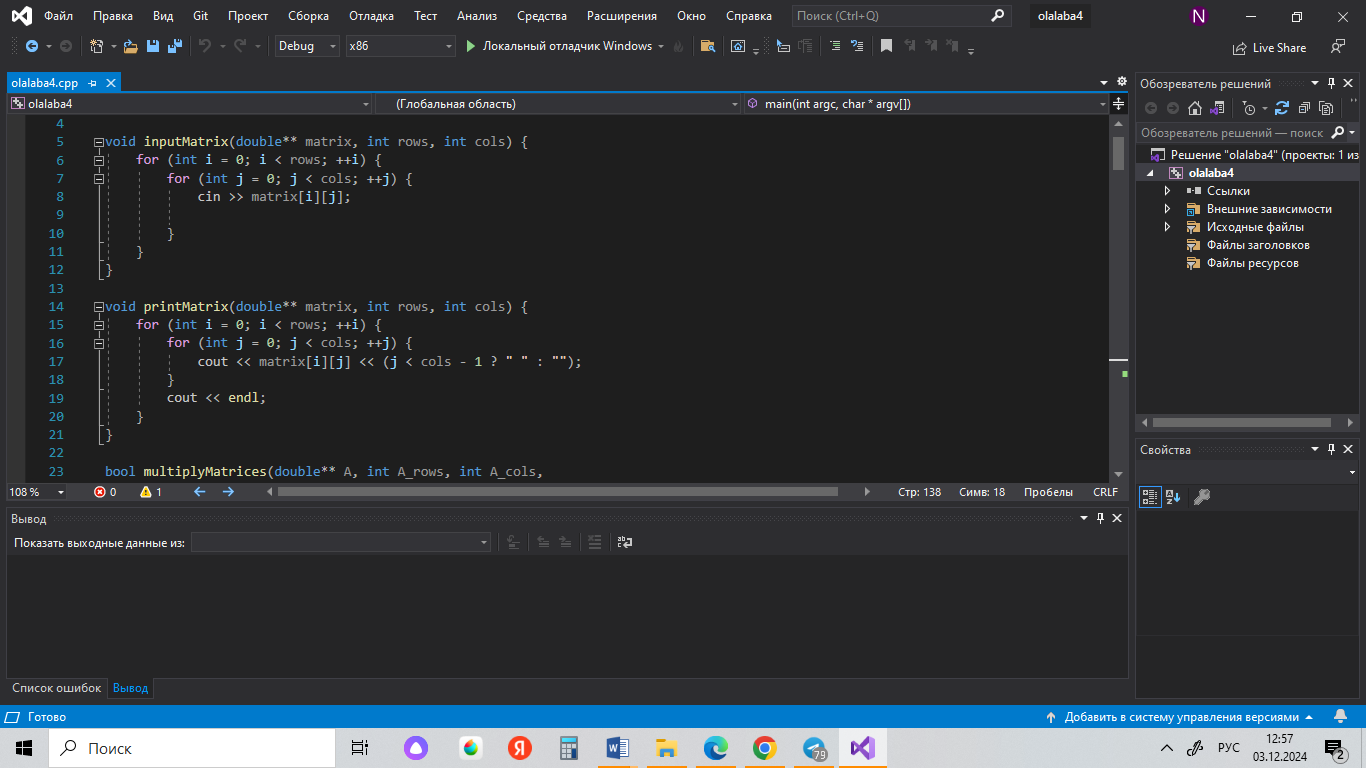


Рисунок 4 – Функция для ввода матрицы

Затем была добавлен функция для вывода матрицы. Эта функция выводит матрицу на экран. Она проходит по всем элементам матрицы и выводит их, добавляя пробел между элементами в строке, но если это последний элемент в строке, то пробел не добавляется.

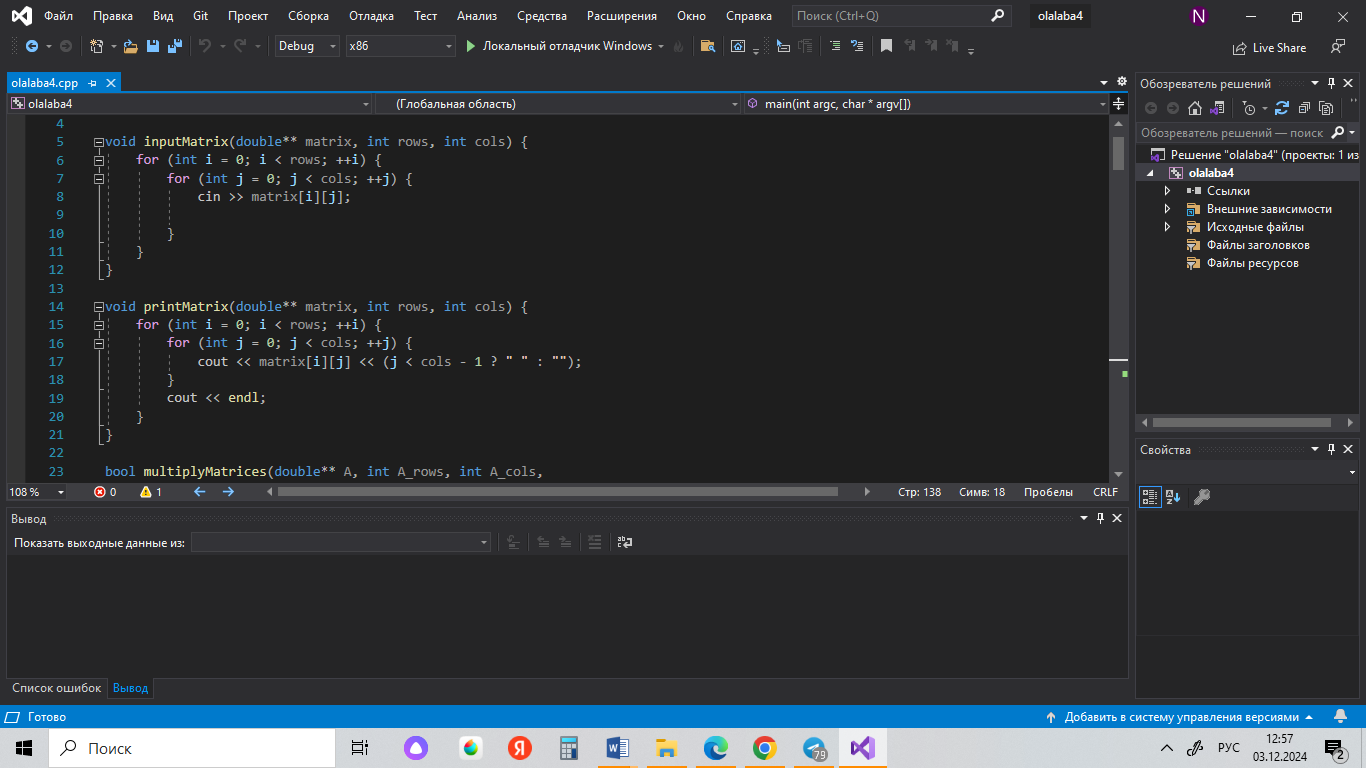


Рисунок 5 – Функция для вывода матрицы

Далее была написана функция для умножения матриц. Эта функция умножает две матрицы A и B.

A – это указатель на первую матрицу. A\_rows и A\_cols – это количество строк и столбцов в матрице A. B – это указатель на вторую матрицу. B\_rows и B\_cols – это количество строк и столбцов в матрице B. result – это ссылка на указатель на результирующую матрицу (будет создана внутри функции). result\_rows и result\_cols – это ссылки для возврата размеров результирующей матрицы.

Далее следует проверка на возможность умножения. Условие проверяет, возможно ли умножение матриц. Условие умножения матриц заключается в том, чтобы количество столбцов первой матрицы (A) должно совпадать с количеством строк второй матрицы (B). Если они не равны, функция возвращает false.

Далее идёт выделение памяти для результирующей матрицы. Используя оператор new, выделяется память под массив указателей на строки для результирующей матрицы. Затем для каждой строки выделяется память под массив элементов (столбцов) матрицы B.

После этого идёт умножение матриц. Внешние циклы i и j проходят по всем индексам результирующей матрицы, инициализируя каждый элемент нулём. Внутренний цикл k выполняет умножение соответствующих элементов из матриц A и B, суммируя их для построения элемента результата result[i][j]. (Выражение A[i][k] \* B[k][j] означает, что берется элемент из i-й строки матрицы A и элемент из j-го столбца матрицы B, которые соответствуют друг другу по индексу k. Это точка, где происходит умножение.)

Далее следует установка размеров результирующей матрицы. После успешного выполнения умножения устанавливаются размеры результирующей матрицы в переменных result\_rows и result\_cols. Функция возвращает true, указывая на успешное выполнение операции.

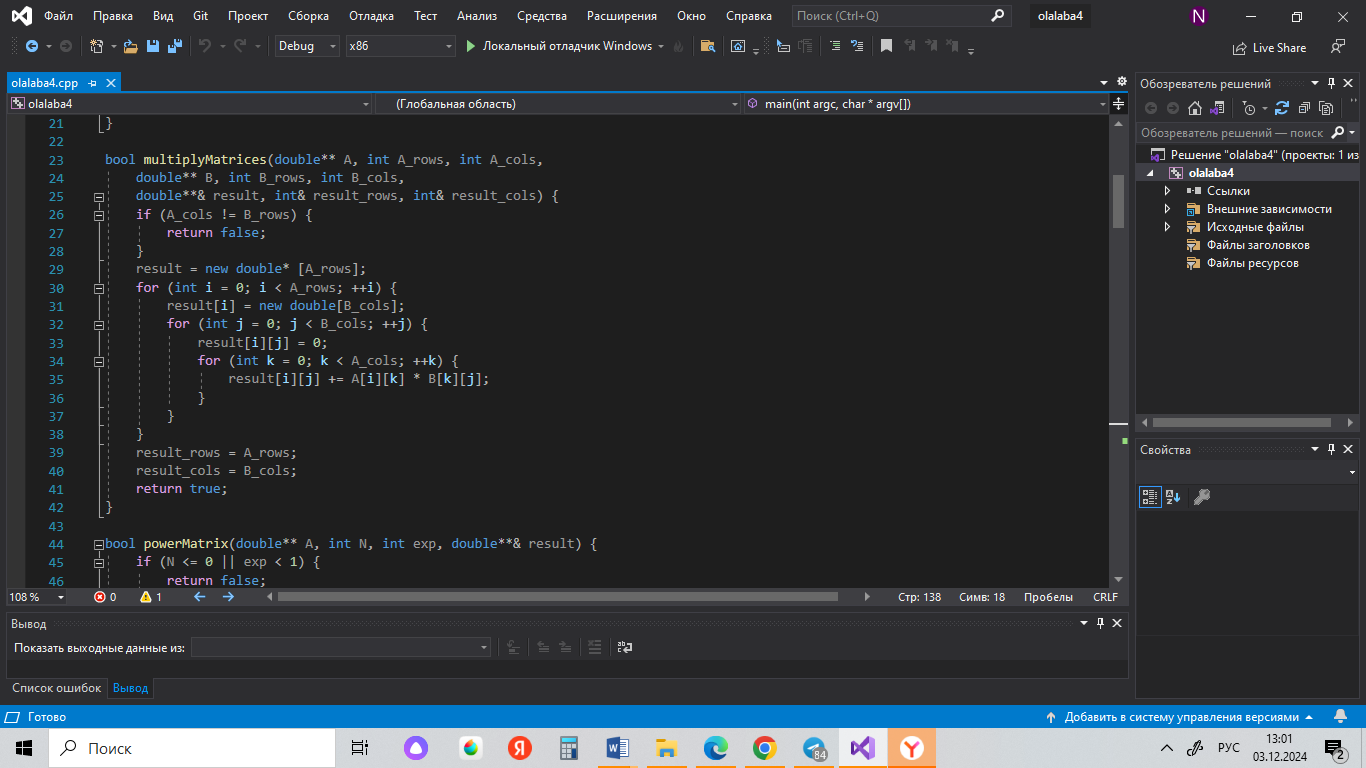


Рисунок 6 – Функция для умножения матриц

Была написана функция для возведения матрицы в степень. Эта функция возводит квадратную матрицу A в степень exp. Если размер матрицы или степень некорректны, возвращается false. Были использованы следующие параметры функции: double\*\* A – это указатель на двумерный массив (в данном случае, на квадратную матрицу типа double), которую мы хотим возвести в степень. int N – это размерность матрицы A (матрица имеет размер N x N). int exp – это степень, в которую мы хотим возвести матрицу A. Она должна быть больше или равна 1. double\*\*& result – это ссылка на указатель. Это означает, что мы будем изменять указатель result в функции, чтобы он указывал на новую матрицу, которая будет хранить результат.

Сначала была проведена проверка входных параметров. Проверяется, что размер матрицы N должен быть положительным и степень exp должна быть не меньше 1. Если это не так, функция возвращает false.

Было написано выделение памяти для результирующей матрицы. Создан новый указатель result, который содержит массив указателей на строки, а затем для каждой строки выделена память под массив типа double. Итоговая матрица имеет размер N x N.

Следующим шагом идёт инициализация результирующей матрицы. Результирующая матрица инициализируется значениями из матрицы A. Так начинается возведение в степень с первого множителя.

Написан цикл для возведения матрицы в степень. Начинаем с 1 и продолжаем до exp - 1. На каждой итерации вызывается функция multiplyMatrices, которая принимает текущую result, матрицу A и результат сохраняет в матрице temp. Если умножение не удалось, функция возвращает false.

Далее идёт копирование результата. После успешного умножения мы копируем результат из временной матрицы temp обратно в result.

Дальше идёт освобождение памяти. После завершения работы с временной матрицей освобождаем память, выделенную под нее.

Последним этапом был написан возврат результата. Если все прошло успешно, функция возвращает true, что указывает на успешное завершение операции возведения в степень.

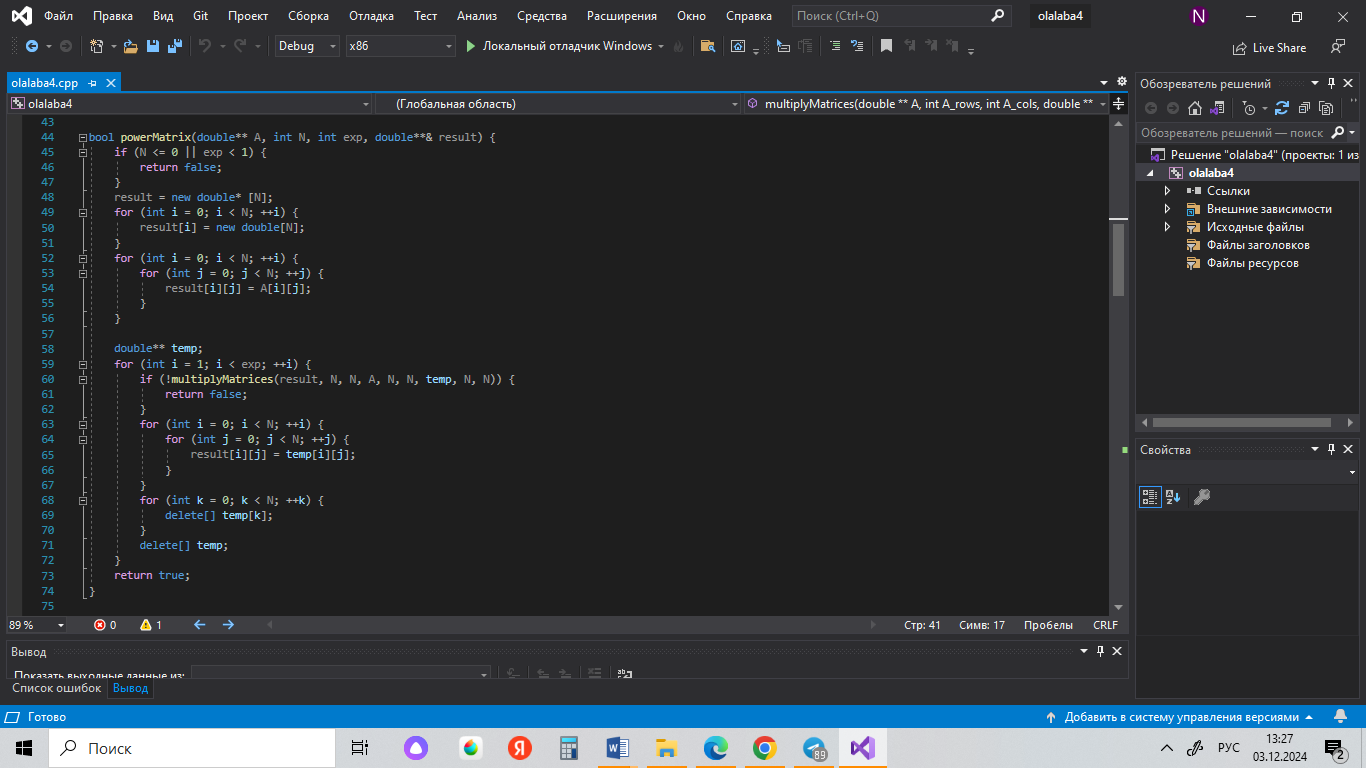


Рисунок 7 – Функция для возведения матрицы в степень

Создано тело функции. Добавлена функция setlocale. Она позволяет устанавливать различные параметры в зависимости от географического положения пользователя, например, позволяет пользоваться тем алфавитом и временем, которые есть в этой стране. Для моей программы было указано “RU”. Также ещё раз использована функция bool для проверки того, чтобы узнать человек пользуется данной программой или робот. Если программу использует человек, для него выводится полный интерфейс с подробным описанием действий. Упрощённый интерфейс краткий и ориентирован на автоматизированную проверку.

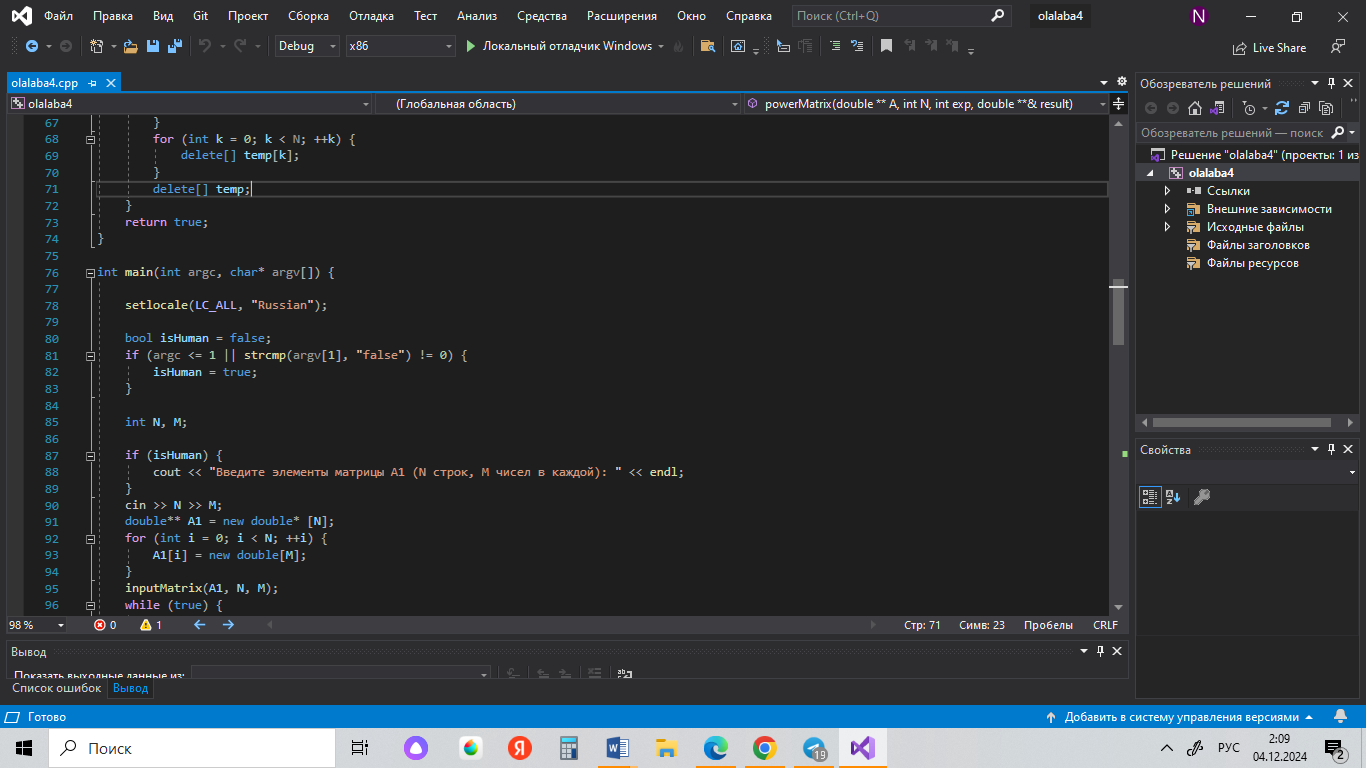


Рисунок 8 – Создание тела функции

Далее был прописан ввод матрицы А1. Объявлены две переменные, N и M, которые будут использоваться для хранения значений матрицы. Для отображения данных в консоли использован оператор cout, который служит для вывода простого текста, а также переменных в консоль. Для получения данных от пользователя использован оператор cin.

Далее была выделена память для массива указателей. A1 — это указатель на указатель типа double (то есть double\*\*), который будет представлять собой двумерный массив (или матрицу) переменной величины. Выделена память для массива указателей с размером N. Каждый элемент этого массива будет указывать на строку в матрице.

Была выделена память для строк матрицы. Цикл for проходит от 0 до N-1. На каждой итерации выделяется память для M элементов типа double, что соответствует каждому ряду матрицы. A1[i] будет указывать на массив из M элементов типа double, представляющий строку i матрицы.

Далее вызывается функция inputMatrix, которая отвечает за ввод конкретных чисел в матрицу.

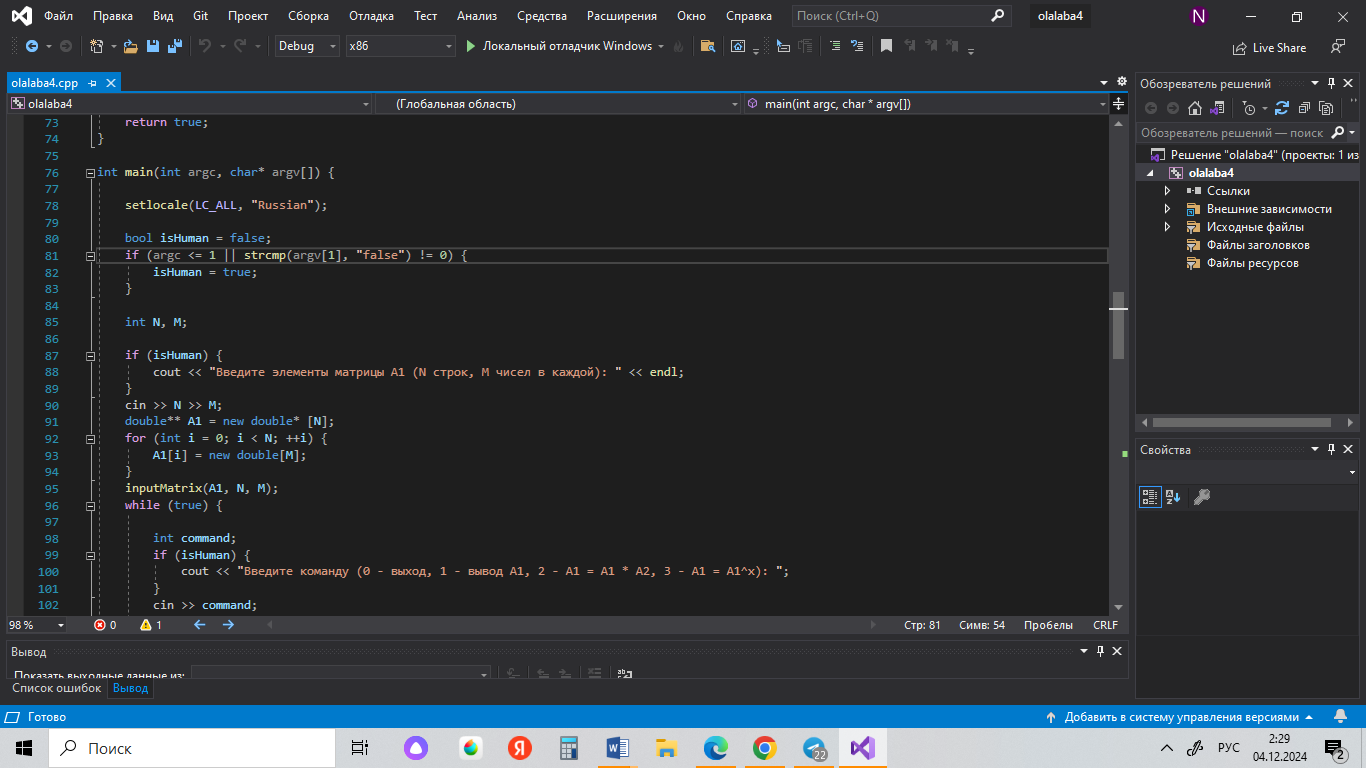


Рисунок 9 – Прописан ввод матрицы А1

Оператор while (true) создает бесконечный цикл, который будет выполняться до тех пор, пока программа не будет прервана. Далее была объявлена переменная command, которая будет использоваться для хранения команды пользователя (0 – выход из программы; 1 – вывести содержимое матрицы A1; 2 – умножить матрицу A1 на другую матрицу A2 и сохранить результат обратно в A1; 3 – возвести матрицу A1 в степень x). Далее происходит ввод команды от пользователя. Функция cin считывает значение, введенное пользователем, и сохраняет его в переменной command.

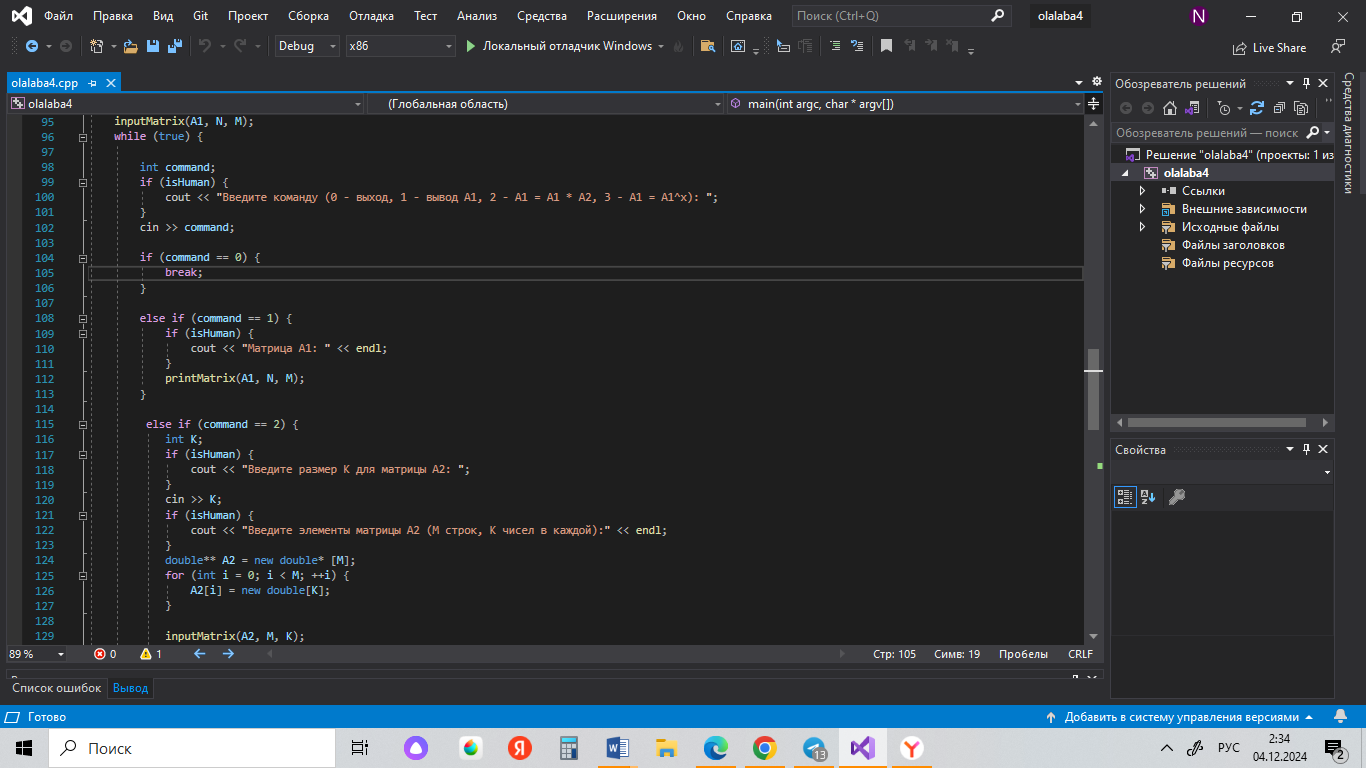


Рисунок 10 – Создан бесконечный цикл

Далее написан код для выхода из программы. Если пользователь ввёл "0", то происходит выход из программы.

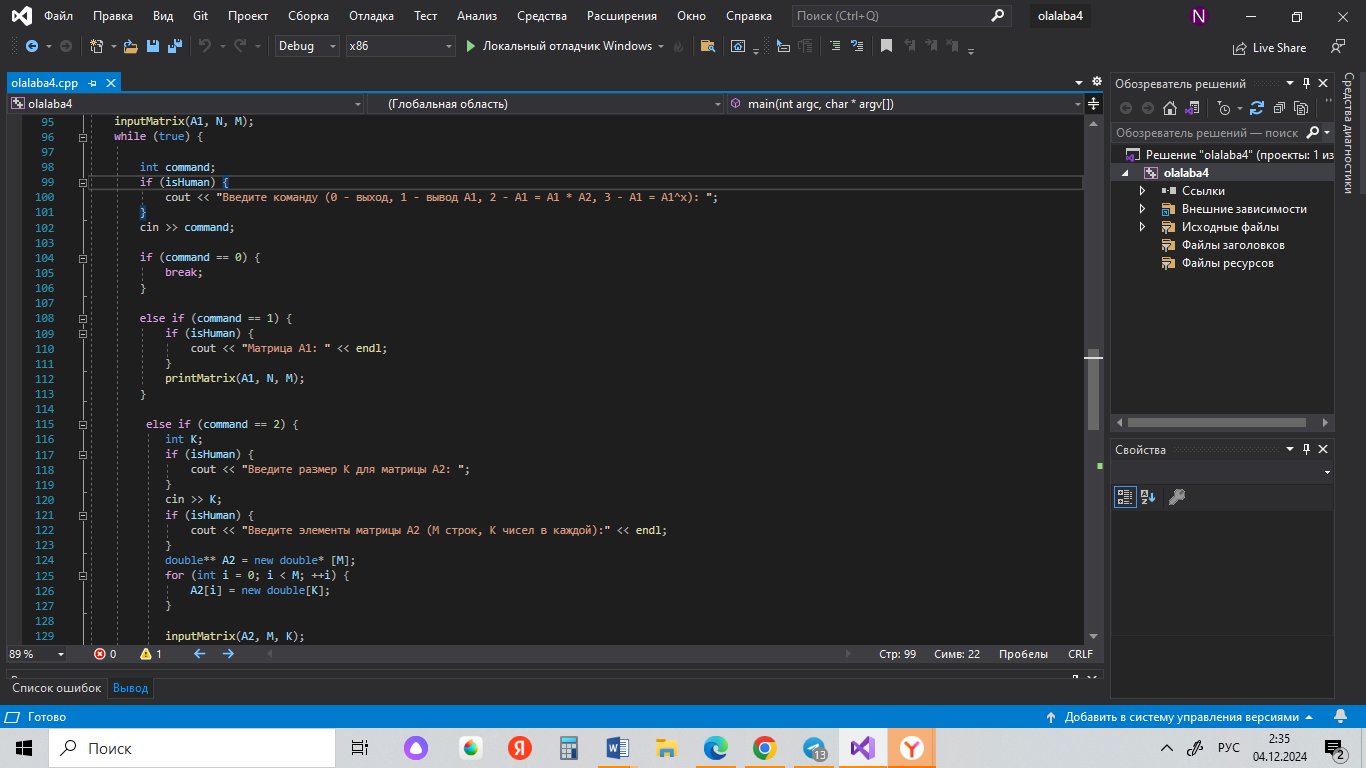


Рисунок 11 – Написан код для выхода из программы

Если пользователь ввел "1", выполнится вызов функции printMatrix, которая отвечает за вывод содержимого матрицы.

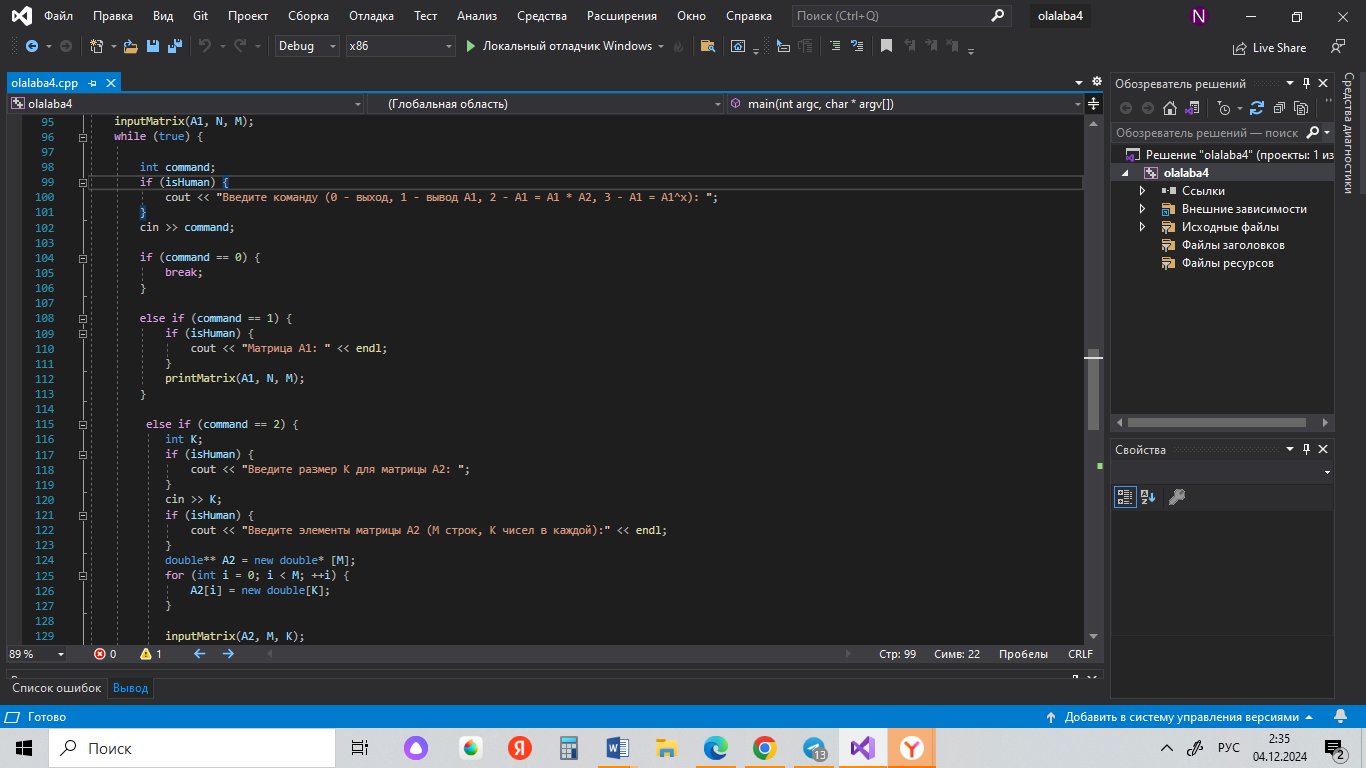


Рисунок 12 – Код для выполнения команды 1

Если пользователь ввел команду "2", значит выполнится умножения матрицы A1 на новую матрицу A2.

Была объявлена переменная K, которая будет использоваться для определения количества столбцов (или элементов) в матрице A2. Далее программа запрашивает у пользователя ввести размер K для матрицы A2, а затем считывает его через cin. После ввода K следует ввод элементов матрицы A2. Выделяется память для матрицы A2. Функция inputMatrix вызывается для считывания значений в матрицу A2. Следующим этапом задаются указатель на результат умножения матриц и переменные для хранения количества строк и столбцов результата.

Далее была вызвана функция multiplyMatrices, которая выполняет умножение матриц A1 и A2. Если умножение не удается (например, если размеры матриц не позволяют выполнить операцию), программа выводит "NO".

Если умножение прошло успешно, программа освобождает память для старой матрицы A1, обновляет её размеры и указывает новый результат на переменную A1. В конце программа освобождает память, выделенную для матрицы A2.

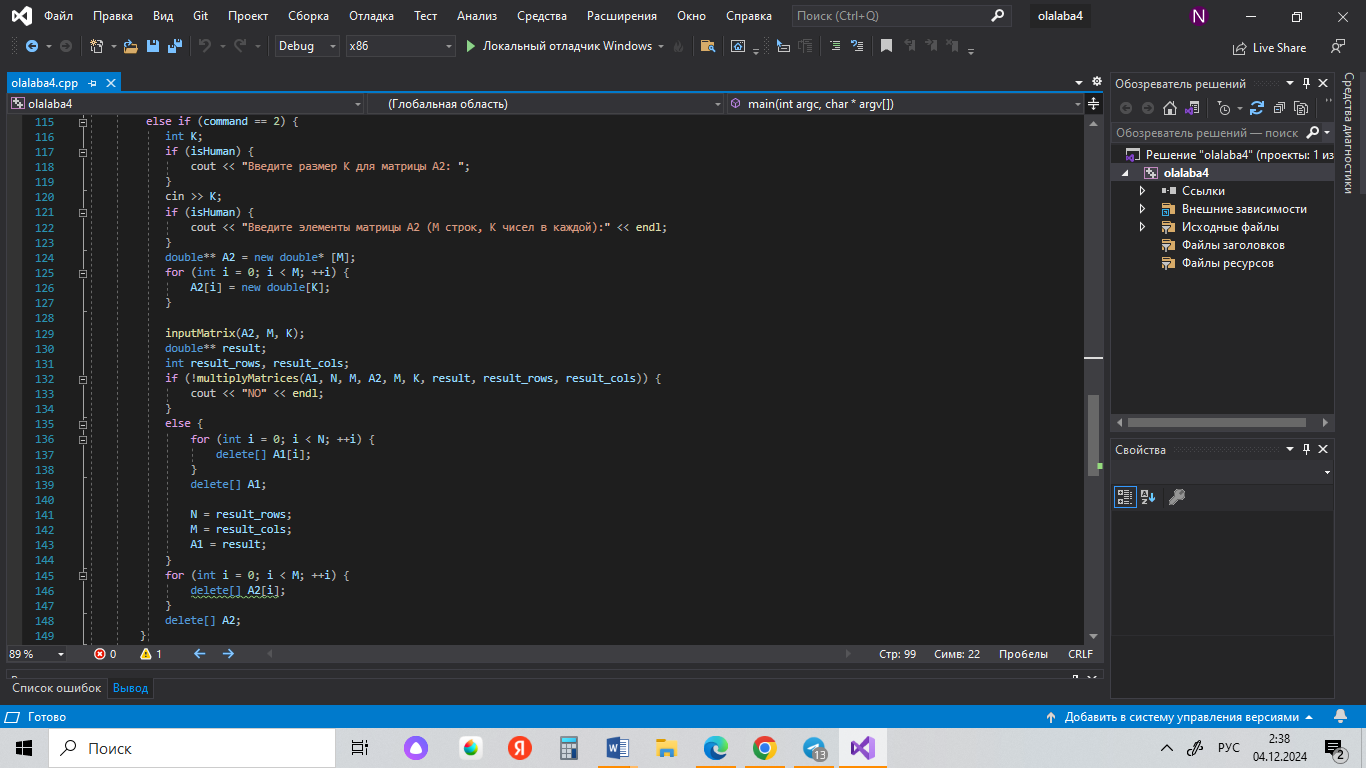


Рисунок 13 – Код для выполнения команды 2

Если пользователь ввел команду "3", значит выполнится возведение матрицы A1 в степень. Сначала выполняется проверка, является ли матрица A1 квадратной, то есть количество строк (N) должно быть равно количеству столбцов (M). Если матрица не квадратная, выводится сообщение "NO" и управление передается к следующей итерации цикла с помощью continue. Это означает, что дальнейшая обработка команды не будет выполнена. Но если матрица квадратная, значит дальше объявляется переменная x, в которой будет храниться натуральное число — степень, в которую будет возведена матрица. Значение x считывается с помощью cin. Далее был объявлен указатель result, который будет использоваться для хранения результирующей матрицы после возведения в степень. После всего была вызвана функция powerMatrix, которая выполняет возведение матрицы A1 в степень x, и результат сохраняется в result. Если возведение в степень невозможно (например, из-за неправильных параметров) функция возвращает false и программа выводит сообщение "NO". Если возведение матрицы в степень прошло успешно (функция powerMatrix вернула true), программа освобождает память, занятую старой матрицей A1.

Потом происходит освобождение памяти для каждой строки матрицы с использованием delete[], затем освобождается сама матрица с помощью delete[]. В конце указатель A1 обновляется для указания на новую матрицу result, полученную в результате возведения в степень.

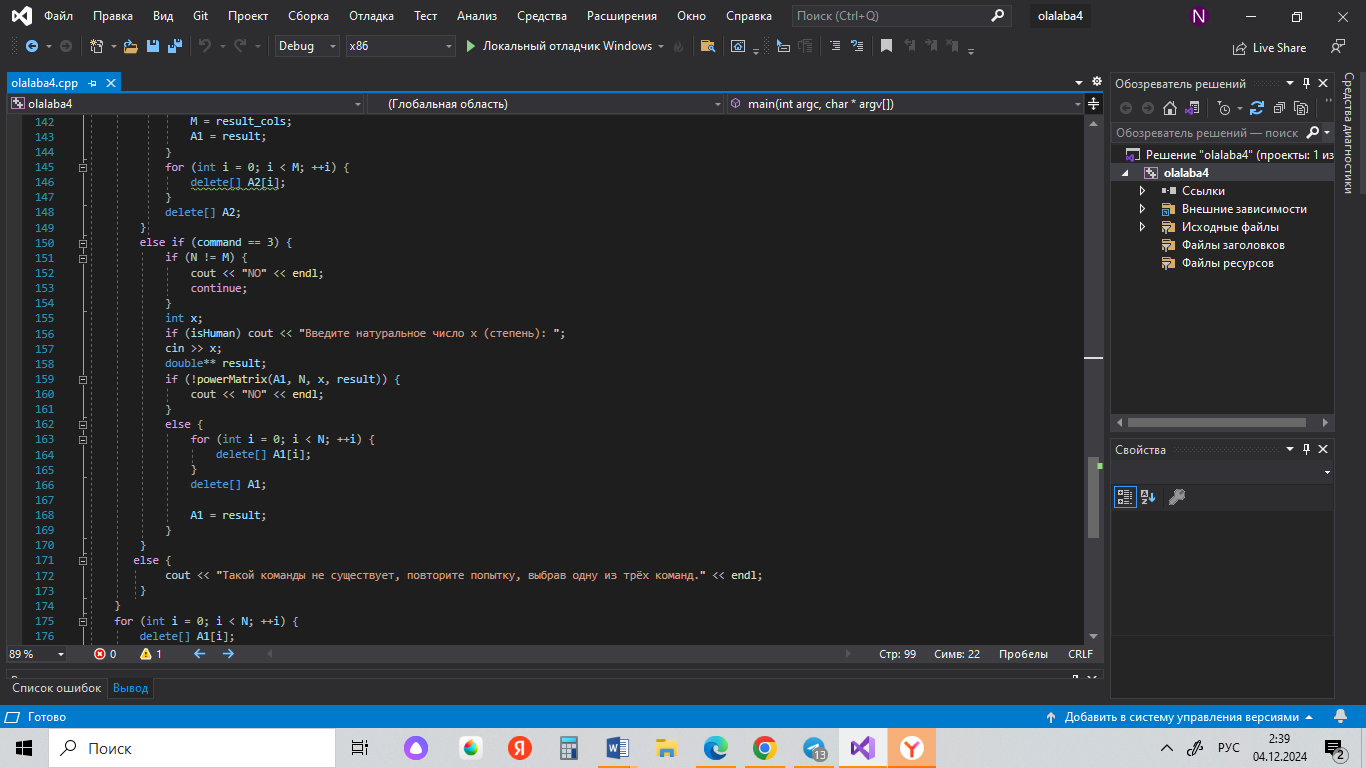


Рисунок 14 – Код для выполнения команды 3

Если пользователь ввёл любую другую команду не из списка команд, то программа выводит предупреждение с предложением повторить попытку, выбрав одну из трёх предложенных команд.

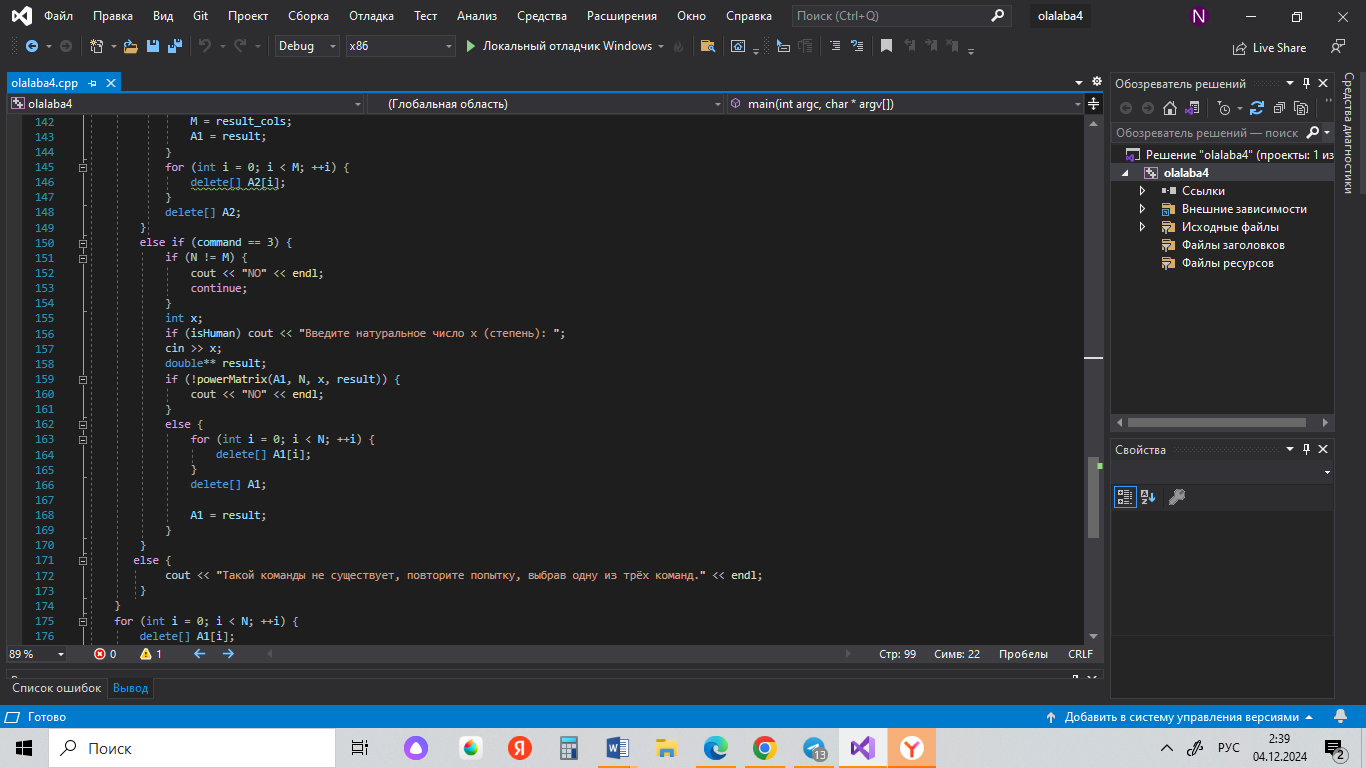


Рисунок 15 – Предупреждение с предложением повторить попытку

В конце программы освобождается память, выделенная под матрицу A1, и программа завершает свою работу.

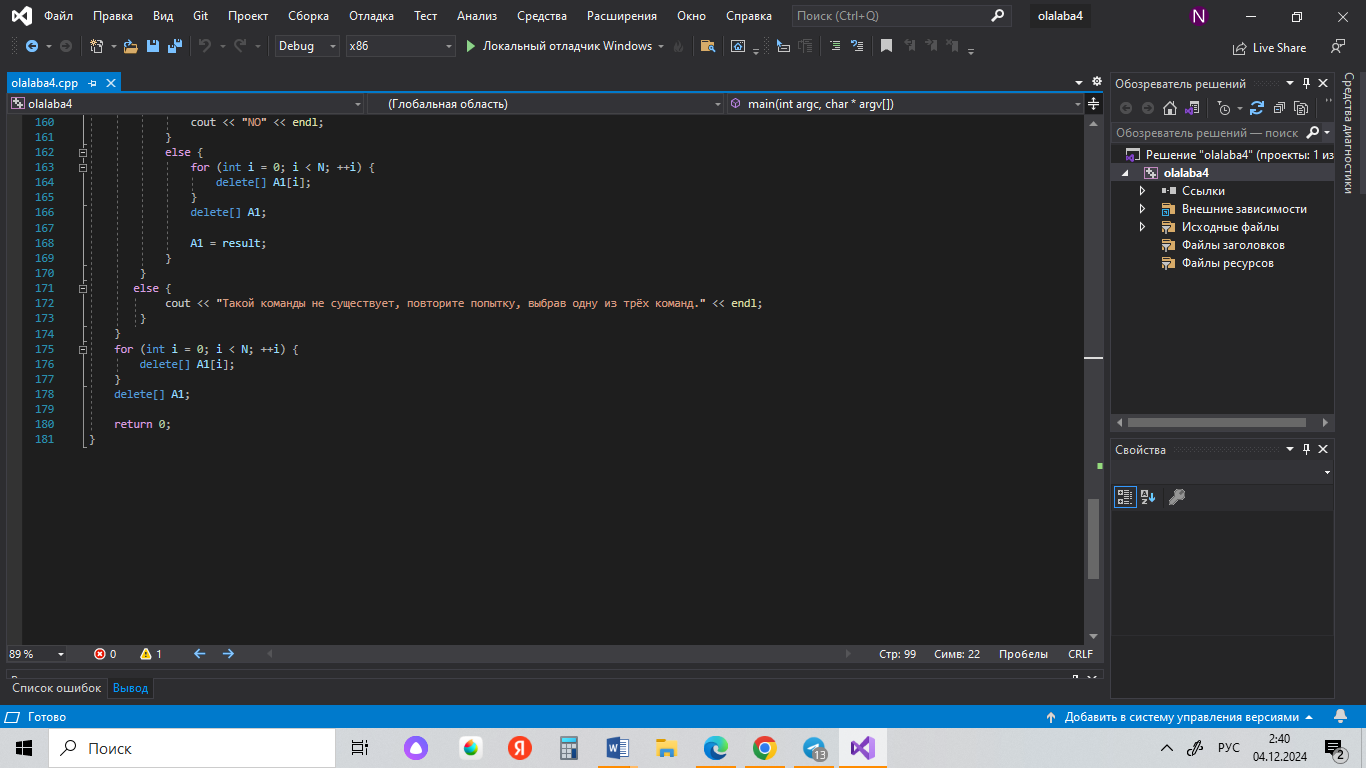


Рисунок 16 – Освобождение памяти

**Тестирование:**

Были введены количество строк и столбцов матрицы, после программа вывела все результаты верно, закончив программу после выбора команды «0».

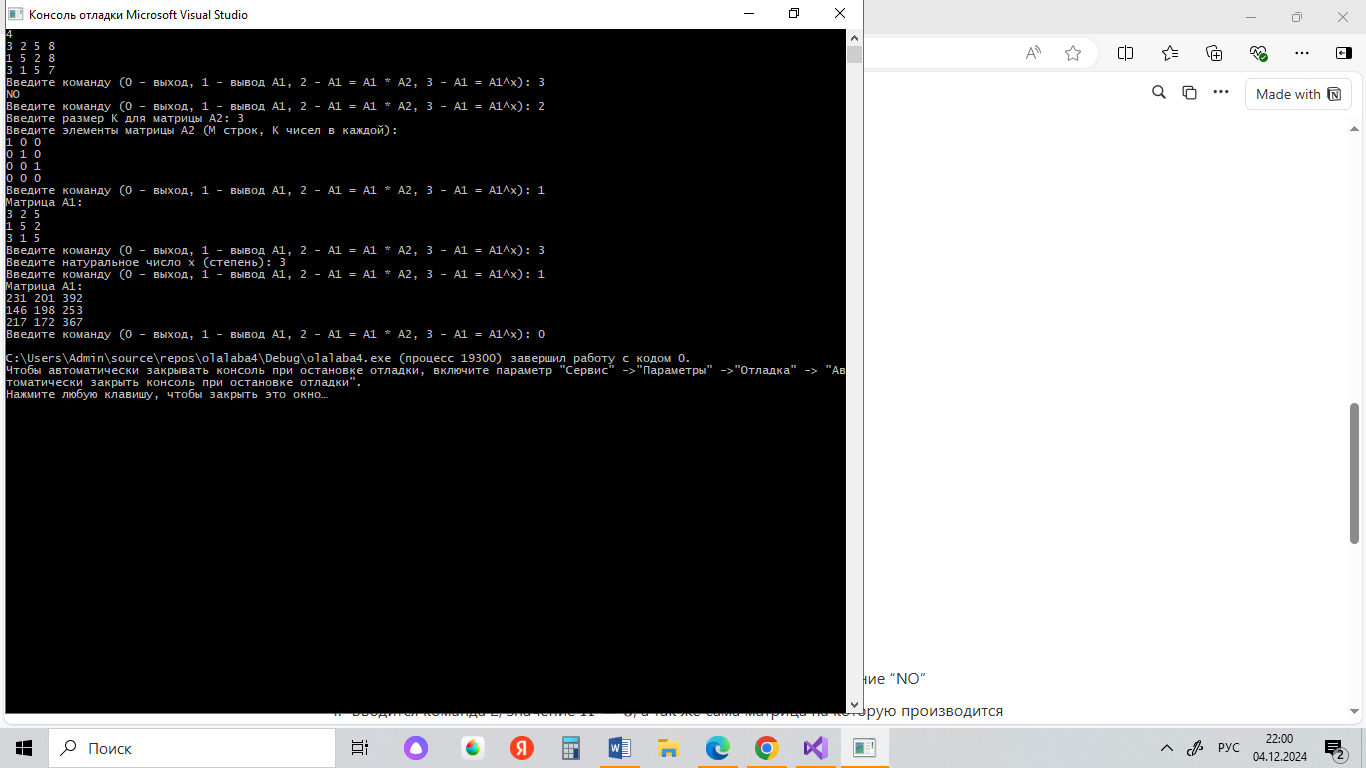


Рисунок 17 – Успешное тестирование

Были введены не числа, а символы и программа сработала не верно из-за неправильного ввода данных.

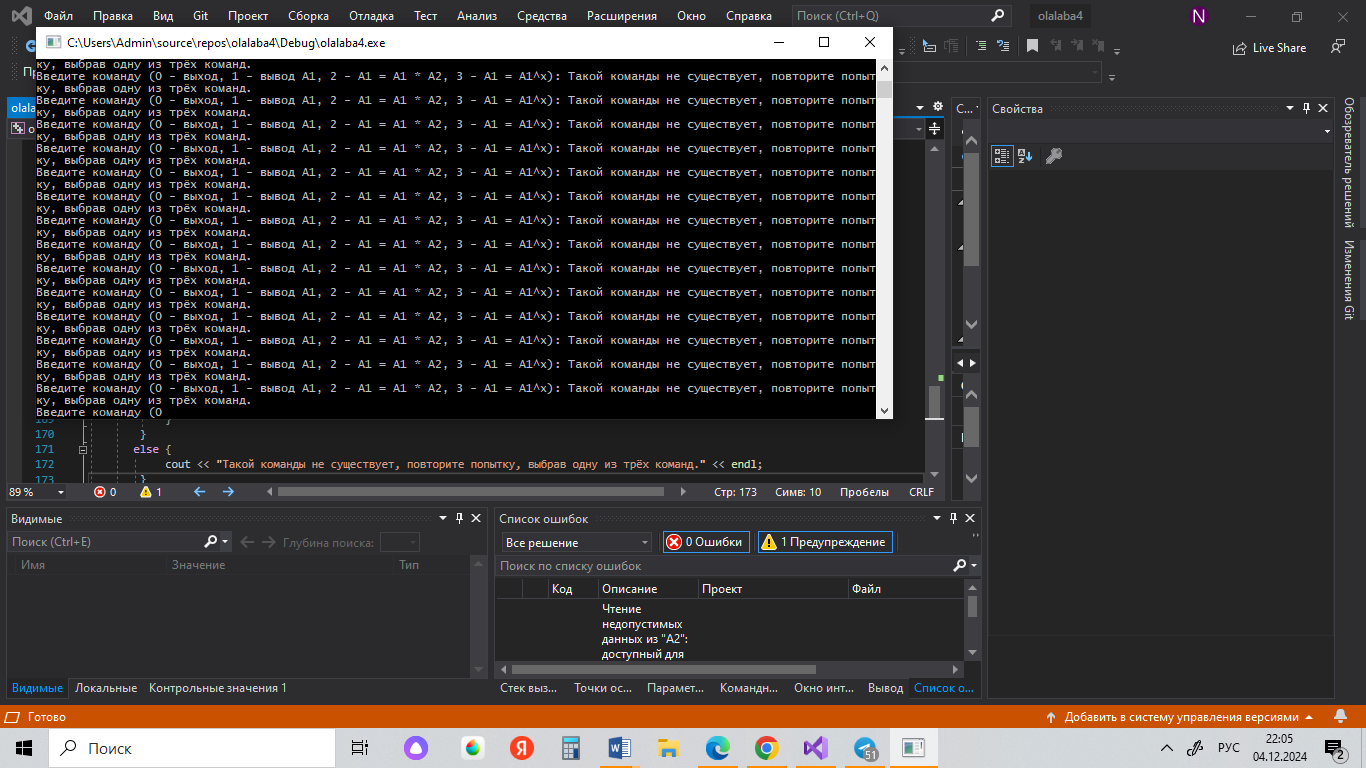


Рисунок 18 – Неправильный ввод данных

**Вывод:**

Было изучено создание двумерных и динамических массивов и работа с ними. Также была написана программа, которая может выводить на экран, умножать и возводить в степень матрицы.

**Код программы:**

#include <iostream>

using namespace std;

void inputMatrix(double\*\* matrix, int rows, int cols) {

for (int i = 0; i < rows; ++i) {

for (int j = 0; j < cols; ++j) {

cin >> matrix[i][j];

}

}

}

void printMatrix(double\*\* matrix, int rows, int cols) {

for (int i = 0; i < rows; ++i) {

for (int j = 0; j < cols; ++j) {

cout << matrix[i][j] << (j < cols - 1 ? " " : "");

}

cout << endl;

}

}

bool multiplyMatrices(double\*\* A, int A\_rows, int A\_cols,

double\*\* B, int B\_rows, int B\_cols,

double\*\*& result, int& result\_rows, int& result\_cols) {

if (A\_cols != B\_rows) {

return false;

}

result = new double\* [A\_rows];

for (int i = 0; i < A\_rows; ++i) {

result[i] = new double[B\_cols];

for (int j = 0; j < B\_cols; ++j) {

result[i][j] = 0;

for (int k = 0; k < A\_cols; ++k) {

result[i][j] += A[i][k] \* B[k][j];

}

}

}

result\_rows = A\_rows;

result\_cols = B\_cols;

return true;

}

bool powerMatrix(double\*\* A, int N, int exp, double\*\*& result) {

if (N <= 0 || exp < 1) {

return false;

}

result = new double\* [N];

for (int i = 0; i < N; ++i) {

result[i] = new double[N];

}

for (int i = 0; i < N; ++i) {

for (int j = 0; j < N; ++j) {

result[i][j] = A[i][j];

}

}

double\*\* temp;

for (int i = 1; i < exp; ++i) {

if (!multiplyMatrices(result, N, N, A, N, N, temp, N, N)) {

return false;

}

for (int i = 0; i < N; ++i) {

for (int j = 0; j < N; ++j) {

result[i][j] = temp[i][j];

}

}

for (int k = 0; k < N; ++k) {

delete[] temp[k];

}

delete[] temp;

}

return true;

}

int main(int argc, char\* argv[]) {

setlocale(LC\_ALL, "Russian");

bool isHuman = false;

if (argc <= 1 || strcmp(argv[1], "false") != 0) {

isHuman = true;

}

int N, M;

if (isHuman) {

cout << "Введите элементы матрицы A1 (N строк, M чисел в каждой): " << endl;

}

cin >> N >> M;

double\*\* A1 = new double\* [N];

for (int i = 0; i < N; ++i) {

A1[i] = new double[M];

}

inputMatrix(A1, N, M);

while (true) {

int command;

if (isHuman) {

cout << "Введите команду (0 - выход, 1 - вывод A1, 2 - A1 = A1 \* A2, 3 - A1 = A1^x): ";

}

cin >> command;

if (command == 0) {

break;

}

else if (command == 1) {

if (isHuman) {

cout << "Матрица А1: " << endl;

}

printMatrix(A1, N, M);

}

else if (command == 2) {

int K;

if (isHuman) {

cout << "Введите размер K для матрицы A2: ";

}

cin >> K;

if (isHuman) {

cout << "Введите элементы матрицы A2 (M строк, K чисел в каждой):" << endl;

}

double\*\* A2 = new double\* [M];

for (int i = 0; i < M; ++i) {

A2[i] = new double[K];

}

inputMatrix(A2, M, K);

double\*\* result;

int result\_rows, result\_cols;

if (!multiplyMatrices(A1, N, M, A2, M, K, result, result\_rows, result\_cols)) {

cout << "NO" << endl;

}

else {

for (int i = 0; i < N; ++i) {

delete[] A1[i];

}

delete[] A1;

N = result\_rows;

M = result\_cols;

A1 = result;

}

for (int i = 0; i < M; ++i) {

delete[] A2[i];

}

delete[] A2;

}

else if (command == 3) {

if (N != M) {

cout << "NO" << endl;

continue;

}

int x;

if (isHuman) cout << "Введите натуральное число x (степень): ";

cin >> x;

double\*\* result;

if (!powerMatrix(A1, N, x, result)) {

cout << "NO" << endl;

}

else {

for (int i = 0; i < N; ++i) {

delete[] A1[i];

}

delete[] A1;

A1 = result;

}

}

else {

cout << "Такой команды не существует, повторите попытку, выбрав одну из трёх команд." << endl;

}

}

for (int i = 0; i < N; ++i) {

delete[] A1[i];

}

delete[] A1;

return 0;

}