-7

Специальность: 09.02.07 Информационные системы и программирование

Выполнил:

Студент группы Y2433

Тертень Д.В.

ОТЧЕТ

о лабораторной работе № 2

по теме: Одномерные и двумерные массивы

по дисциплине: Разработка программных модулей

Санкт-Петербург 2020

Проверил:

Антонов. М.Б.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

УНИВЕРСИТЕТ ИТМО

Факультет среднего профессионального образования

1. Цель и задачи

Цель: Создать программу работающую с двумерными массивами.

Задачи:

* научиться работать с двумерными массивами;
* научиться передавать и возвращать несколько значений функции;
* научиться создавать динамические массивы

1. Задание

Пользователь должен задать размер двух матриц. Ввод с клавиатуры:

* количество строк первой матрицы;
* количество столбцов первой матрицы;
* количество строк второй матрицы;
* количество столбцов второй матрицы.

Требуется реализовать обязательно в виде отдельных функций:

* сумму двух матриц;
* произведение двух матриц;
* транспонирование матрицы;
* сложение и умножение матрицы на число.

Реализовать дополнительно в виде отдельных функций:

* вычисление дискреминанта матрицы;
* нормализацию матрицы;
* оформить вывод матрицы на экран в виде отдельной функции.

Если какое-либо действие невозможно выполнить — написать соответствующее сообщение об ошибке.

Инициализация вывода матрицы можно вызвать только через main.

1. КОд программы

Файл main.cpp:

/\*\*

\* @file main.cpp

\* @brief Лабораторная работа 2

\* @author Terten Daniil

\*/

#include <iostream>

#include "matrix.h"

#include "summatrices.h"

#include "productmatrices.h"

#include "transposematrices.h"

#include "determinantmatrices.h"

#include "normalizedmatrices.h"

using namespace std;

/\*\*

\* @brief Точка входа в программу, получение и вывод всех данных о матрицах

\* @return 0

\*/

int main() {

int x1, y1, x2, y2;

int \*\*Matrix1 = getMatrixInt(x1, y1);

int \*\*Matrix2 = getMatrixInt(x2, y2);

try {

cout << endl << "Sum:" << endl;

int \*\*Sum = getSumOfMatrices(Matrix1, x1, y1, Matrix2, x2, y2);

printMatrix(Sum, x1, y1);

} catch (exception &e) {

cout << e.what() << endl;

}

try {

cout << endl << "Product:" << endl;

int \*\*Product = getProductOfMatrices(Matrix1, x1, y1, Matrix2, x2, y2);

printMatrix(Product, x1, y2);

} catch (exception &e) {

cout << e.what() << endl;

}

cout << endl << "Transposed 1:" << endl;

printMatrix(getTransposedMatrix(Matrix1, x1, y1), y1, x1);

cout << endl << "Transposed 2:" << endl;

printMatrix(getTransposedMatrix(Matrix2, x2, y2), y2, x2);

int number;

cout << endl << "Input number:" << endl;

cin >> number;

cout << endl << "Sum of matrix1 and number:" << endl;

printMatrix(getSumOfMatrixAndNumber(Matrix1, x1, y1, number), x1, y1);

cout << endl << "Product of matrix2 and number:" << endl;

printMatrix(getProductOfMatrixAndNumber(Matrix2, x2, y2, number), x2, y2);

try {

cout << endl << "Determinant of 1st matrix:" << endl;

int determinant = getDeterminantOfMatrix(Matrix1, x1, y1);

cout << determinant << endl;

} catch (exception& e) {

cout << e.what() << endl;

}

try {

cout << endl << "Normalized matrix2:" << endl;

int \*\*Normalized = getNormalizedMatrix(Matrix2, x2, y2);

printMatrix(Normalized, x2, y2);

} catch (exception &e) {

cout << e.what() << endl;

}

return 0;

}

Файл determinantmatrices.h:

/\*\*

\* @file determinantmatrices.h

\* @author Terten Daniil

\* @brief Заголовочный файл для функции детерминанта матрицы

\*/

#ifndef LAB2\_DETERMINANTMATRICES\_H

#define LAB2\_DETERMINANTMATRICES\_H

/\*\*

\* @brief Получить детерминант матрицы

\* @param Matrix Матрица

\* @param x Количество строк

\* @param y Количество столбцов

\* @return Результат

\*/

int getDeterminantOfMatrix(int \*\*Matrix, int x, int y);

#endif //LAB2\_DETERMINANTMATRICES\_H

Файл determinantmatrices.cpp:

/\*\*

\* @file determinantmatrices.cpp

\* @author Terten Daniil

\* @brief Реализация функции детерминанта матрицы

\*/

#include "determinantmatrices.h"

#include <iostream>

using namespace std;

int getDeterminantOfMatrix(int \*\*Matrix, int x, int y)

{

if (x != y) {

throw runtime\_error("Matrix has different X and Y");

}

if (x == 1 && y == 1) {

return Matrix[0][0];

}

int sum = 0;

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

int \*\*MinorMatrix = new int \*[x - 1];

for (int j = 0; j < x - 1; ++j) {

MinorMatrix[j] = new int [y - 1];

for (int k = 0; k < y - 1; ++k) {

int curX, curY;

curX = j + 1;

curY = (k < i) ? (k) : (k + 1);

MinorMatrix[j][k] = Matrix[curX][curY];

}

}

int minus = (i % 2) ? -1 : 1;

sum += minus \* Matrix[0][i] \* getDeterminantOfMatrix(MinorMatrix, x - 1, y - 1);

}

return sum;

}

Файл matrix.h:

/\*\*

\* @file matrix.h

\* @author Terten Daniil

\* @brief Заголовочный файл для функций по работе с матрицами

\*/

#ifndef LAB2\_MATRIX\_H

#define LAB2\_MATRIX\_H

/\*\*

\* @brief Функция заполнения матрицы

\* @param Matrix Матрица для заполнения

\* @param x Кол-во строк

\* @param y Кол-во столбцов

\* @return Результат

\*/

int \*\*fillMatrix(int \*\*Matrix, int x, int y);

/\*\*

\* @brief Cоздание матрицы заданных размеров

\* @param x Кол-во строк

\* @param y Кол-во столбцов

\* @return Указатель на матрицу

\*/

int \*\*getMatrixInt(int &x, int &y);

/\*\*

\* @brief Вывод матрицы на экран

\* @param Matrix Матрица для вывода

\* @param x Кол-во строк

\* @param y Кол-во столбцов

\*/

void printMatrix(int \*\*Matrix, int x, int y);

#endif //LAB2\_MATRIX\_H

Файл matrix.cpp:

/\*\*

\* @file matrix.cpp

\* @author Terten Daniil

\* @brief Реализация функций для работы с матрицами

\*/

#include "matrix.h"

#include <iostream>

using namespace std;

int \*\*fillMatrix(int \*\*Matrix, int x, int y)

{

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

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

cin >> Matrix[i][j];

}

}

return Matrix;

}

int \*\*getMatrixInt(int &x, int &y)

{

cout << "Input matrix sizes: ROWS, COLUMNS" << endl;

cin >> x >> y;

int \*\*Matrix = new int \*[x];

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

Matrix[i] = new int [y];

}

cout << "Fill matrix:" << endl;

return fillMatrix(Matrix, x, y);

}

void printMatrix(int \*\*Matrix, int x, int y)

{

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

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

cout << Matrix[i][j] << " ";

}

cout << endl;

}

}

Файл normalizedmatrices.h:

/\*\*

\* @file normalizedmatrices.h

\* @author Terten Daniil

\* @brief Заголовочный файл для функции нормализации

\*/

#ifndef LAB2\_NORMALIZEDMATRICES\_H

#define LAB2\_NORMALIZEDMATRICES\_H

/\*\*

\* @brief Получить нормализованную матрицу

\* @param Matrix Матрица для нормализации

\* @param x Кол-во строк

\* @param y Кол-во столбцов

\* @return Указатель на результат

\*/

int \*\*getNormalizedMatrix(int \*\*Matrix, int x, int y);

#endif //LAB2\_NORMALIZEDMATRICES\_H

Файл normalizedmatrices.cpp:

/\*\*

\* @file normalizedmatrices.cpp

\* @author Terten Daniil

\* @brief Реализация функции для нормализации

\*/

#include "normalizedmatrices.h"

#include <iostream>

int \*\*getNormalizedMatrix(int \*\*Matrix, int x, int y)

{

if (x != y) {

throw std::runtime\_error("Matrix has different X and Y");

}

int \*\*Normalized = new int \*[x];

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

Normalized[i] = new int [y];

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

Normalized[i][j] = Matrix[i][j] / Matrix[i][i];

}

}

return Normalized;

}

Файл productmatrices.h:

/\*\*

\* @file productmatrices.h

\* @author Terten Daniil

\* @brief Заголовочный файл для функций сумм для матриц

\*/

#ifndef LAB2\_PRODUCTMATRICES\_H

#define LAB2\_PRODUCTMATRICES\_H

/\*\*

\* @brief Произведение матриц

\* @param Matrix1 Первая матрица

\* @param x1 Кол-во строк первой матрицы

\* @param y1 Кол-во столбцов первой матрицы

\* @param Matrix2 Вторая матрица

\* @param x2 Кол-во строк второй матрицы

\* @param y2 Кол-во столбцов второй матрицы

\* @return Указатель на произведение матриц

\*/

int \*\*getProductOfMatrices(int \*\*Matrix1, int x1, int y1, int \*\*Matrix2, int x2, int y2);

/\*\*

\* @brief Произведение матрицы и числа

\* @param Matrix Матриц

\* @param x Кол-во строк

\* @param y Кол-во столбцов

\* @param number Число

\* @return Указатель на произведение

\*/

int \*\*getProductOfMatrixAndNumber(int \*\*Matrix, int x, int y, int number);

#endif //LAB2\_PRODUCTMATRICES\_H

Файл productmatrices.cpp:

/\*\*

\* @file productmatrices.cpp

\* @author Terten Daniil

\* @brief Реализация функций для произведений матриц

\*/

#include "productmatrices.h"

#include <iostream>

using namespace std;

int \*\*getProductOfMatrices(int \*\*Matrix1, int x1, int y1, int \*\*Matrix2, int x2, int y2)

{

if (y1 != x2) {

throw runtime\_error("Matrices have different Y1 and X2");

}

int \*\*Product = new int \*[x1];

for (int i = 0; i < x1; i++) {

Product[i] = new int [y2];

for (int j = 0; j < y2; j++) {

Product[i][j] = 0;

for (int k = 0; k < y1; k++) {

Product[i][j] += Matrix1[i][k] \* Matrix2[k][j];

}

}

}

return Product;

}

int \*\*getProductOfMatrixAndNumber(int \*\*Matrix, int x, int y, int number)

{

int \*\*Product = new int \*[x];

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

Product[i] = new int [y];

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

Product[i][j] = Matrix[i][j] \* number;

}

}

return Product;

}

Файл summatrices.h:

/\*\*

\* @file summatrices.h

\* @author Terten Daniil

\* @brief Заголовочный файл для функций сумм матриц и чисел

\*/

#ifndef LAB2\_SUMMATRICES\_H

#define LAB2\_SUMMATRICES\_H

/\*\*

\* @brief Сумма матриц

\* @param Matrix1 Первая матрица

\* @param x1 Кол-во строк первой матрицы

\* @param y1 Кол-во столбцов первой матрицы

\* @param Matrix2 Вторая матрица

\* @param x2 Кол-во строк второй матрицы

\* @param y2 Кол-во столбцов второй матрицы

\* @return Указатель на произведение

\*/

int \*\*getSumOfMatrices(int \*\*Matrix1, int x1, int y1, int \*\*Matrix2, int x2, int y2);

/\*\*

\* @brief Сумма матрицы и числа

\* @param Matrix Матриц

\* @param x Кол-во строк

\* @param y Кол-во столбцов

\* @param number Число

\* @return Указатель на сумму

\*/

int \*\*getSumOfMatrixAndNumber(int \*\*Matrix, int x, int y, int number);

#endif //LAB2\_SUMMATRICES\_H

Файл summatrices.cpp:

/\*\*

\* @file summatrices.cpp

\* @author Terten Daniil

\* @brief Реализация функций суммы матриц

\*/

#include "summatrices.h"

#include <iostream>

using namespace std;

int \*\*getSumOfMatrices(int \*\*Matrix1, int x1, int y1, int \*\*Matrix2, int x2, int y2)

{

if (x1 != x2 || y1 != y2) {

throw runtime\_error("Matrices have different sizes");

}

int \*\*Sum = new int \*[x1];

for (int i = 0; i < x1; ++i) {

Sum[i] = new int [y1];

for (int j = 0; j < y1; ++j) {

Sum[i][j] = Matrix1[i][j] + Matrix2[i][j];

}

}

return Sum;

}

int \*\*getSumOfMatrixAndNumber(int \*\*Matrix, int x, int y, int number)

{

int \*\*Sum = new int \*[x];

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

Sum[i] = new int [y];

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

Sum[i][j] = Matrix[i][j] + number;

}

}

return Sum;

}

Файл transposematrices.h:

/\*\*

\* @file transposematrices.h

\* @author Terten Daniil

\* @brief Заголовочный файл для функции транспонирования матрицы

\*/

#ifndef LAB2\_TRANSPOSEMATRICES\_H

#define LAB2\_TRANSPOSEMATRICES\_H

/\*\*

\* @brief Транспонирование матрицы

\* @param Matrix Матрицы для транспонирования

\* @param x Кол-во строк

\* @param y Кол-во столбцов

\* @return Указатель на результат

\*/

int \*\*getTransposedMatrix(int \*\*Matrix, int x, int y);

#endif //LAB2\_TRANSPOSEMATRICES\_H

Файл transposematrices.cpp:

/\*\*

\* @file transposematrices.cpp

\* @author Terten Daniil

\* @brief Реализация функции транспонирования матрицы

\*/

#include "transposematrices.h"

int \*\*getTransposedMatrix(int \*\*Matrix, int x, int y)

{

int \*\*Transposed = new int \*[y];

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

Transposed[i] = new int [x];

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

Transposed[i][j] = Matrix[j][i];

}

}

return Transposed;

}

В качестве результата работы программы представлены скриншоты результатов выполнения программы на рисунках 1-2.

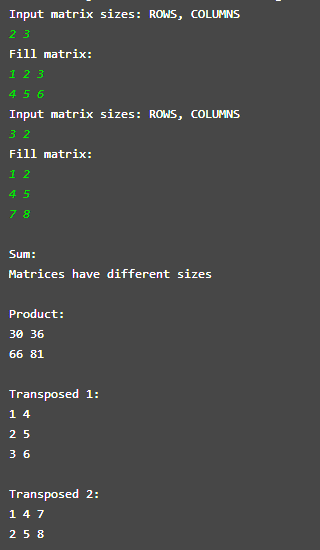


Рисунок 1 – ввод матриц и вывод результатов расчётов

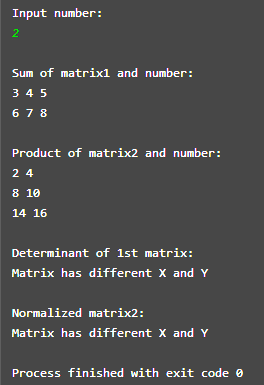


Рисунок 2 – вывод результатов расчётов

1. Диаграммы активности

Диаграммы активности представлены на рисунках 3, 4.

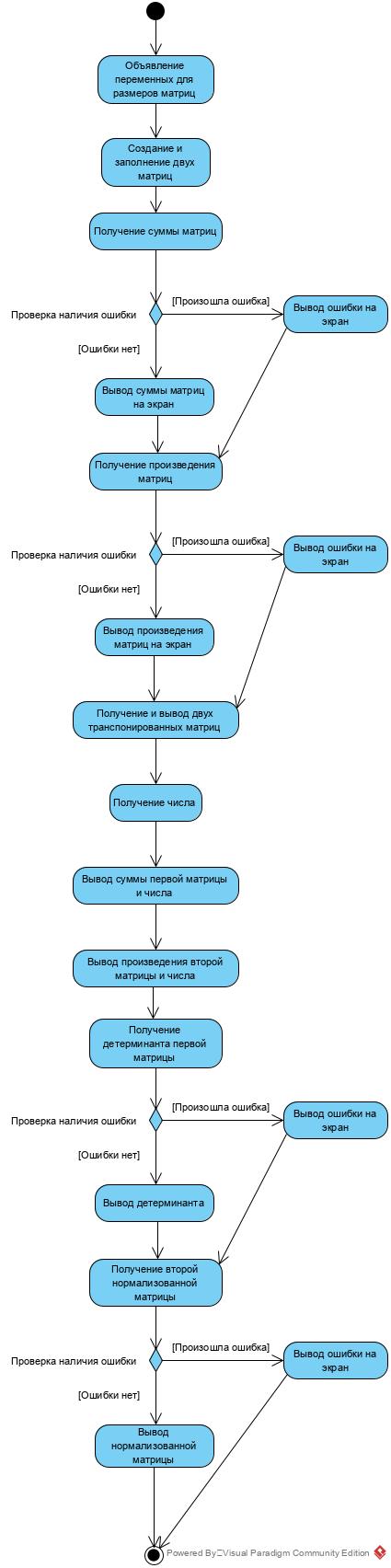


Рисунок 3 – диаграмма функции main

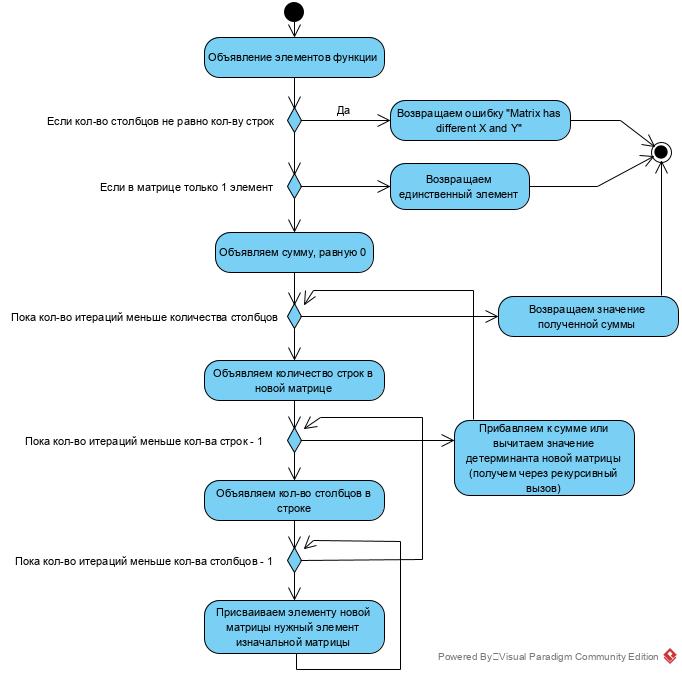


Рисунок 4 – диаграмма функции getDeterminantOfMatrix

# ВЫВОД

В процессе выполнения лабораторной работы были получены навыки работы с двумерными массивами, разделением программы на файлы, передачи параметров по указателям и ссылкам.