**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №3**

**по дисциплине «Программирование»**

Тема: Работа с числовыми матрицами

|  |  |  |
| --- | --- | --- |
| Студентгр. 9893 |  | Росихин А.Ю. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2019

**Цель работы.**

Научиться работать с числовыми матрицами на языке программирования С++. Научиться вычислять определитель и ранг матрицы. Научиться загружать матрицы из файла. Научиться взаимодействовать с динамическим массивом числовых данных.

**Основные теоретические положения.**

Работа с файлами. Для работы с файлами необходимо подключить заголовочный файл <fstream>. В <fstream> определены несколько классов и подключены заголовочные файлы <ifstream> — файловый ввод [1].

Функция getline() имеетконструкцию getline(stream, string, separator),где stream — это поток данных, string — переменная, в которую запишется строка и separator – строковый разделитель, показывающий на конец строки [2]. С помощью функции file.eof() проверяем окончание файла.

Определи́тель (или детермина́нт) — одно из основных понятий линейной алгебры. Определитель квадратной матрицы *A* размеров *N x N*, заданной над коммутативным кольцом *R*, является элементом кольца *R*, вычисляемым по формуле, приведённой ниже [3]. Для матрицы *2 х 2* определитель вычисляется как см. рис. 1.

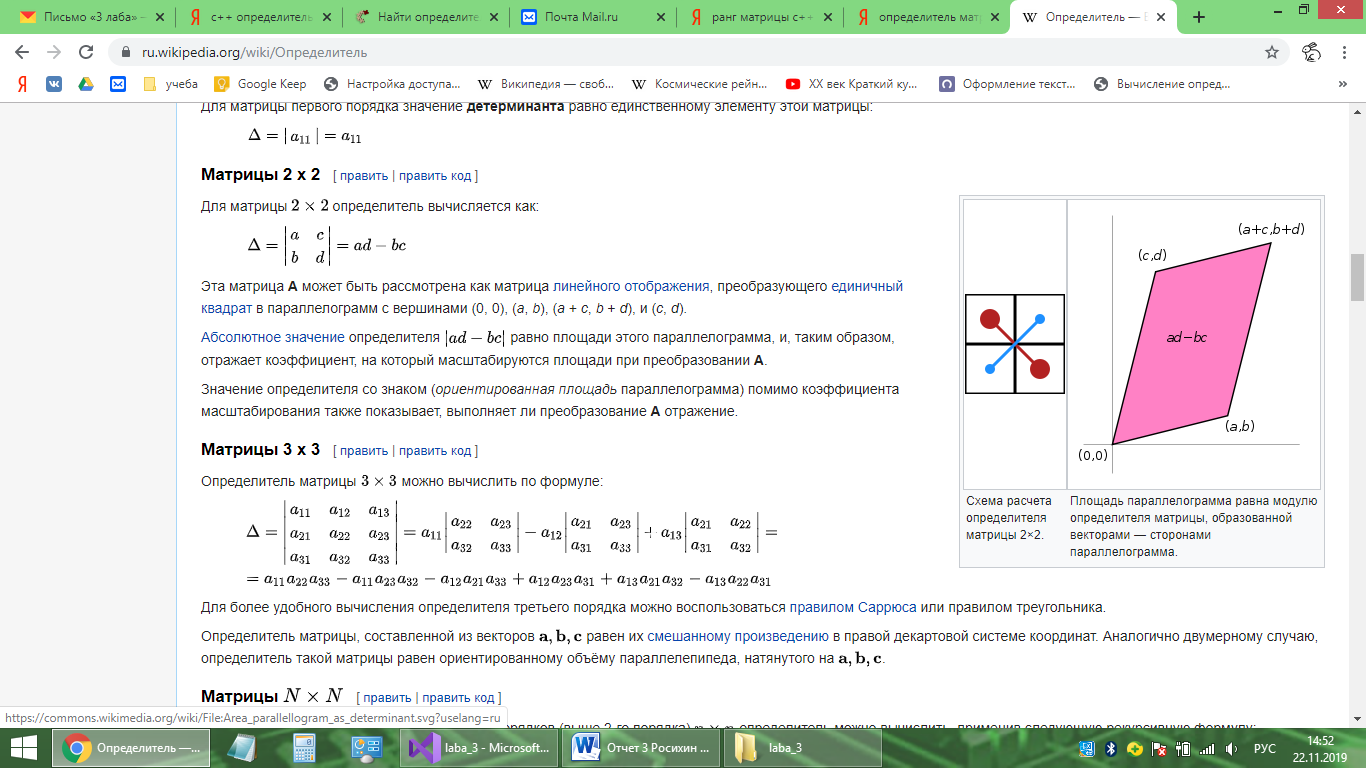
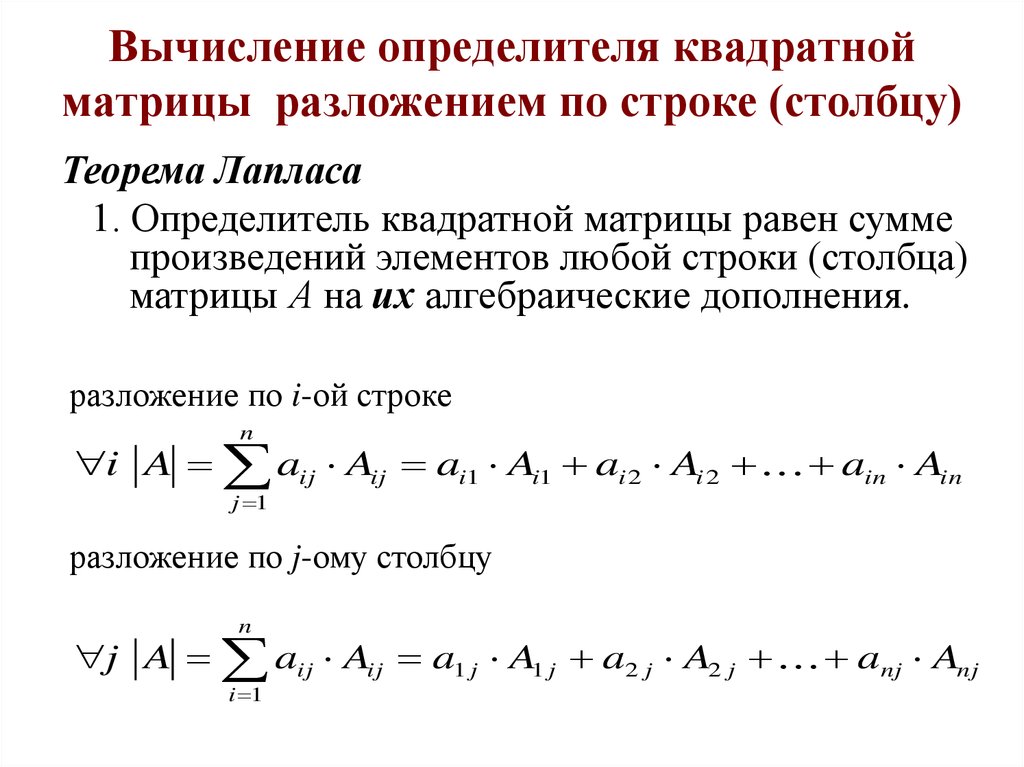


Рисунок 1 — Вычисление определителя матрицы 2 х 2

Если матрица размера больше чем *2 х 2*, тогда мы приводим ее к виду *2 х 2* с помощью рекурсивных функций. Далее определитель мы можем вычислить, применяя теорему Лапласа см. рис. 2.

Рисунок 2 — Теорема Лапласа

Магический, или волшебный квадрат — квадратная таблица *N x N*, заполненная различными числами таким образом, что сумма чисел в каждой строке, каждом столбце и на обеих диагоналях одинакова [4].

**Постановка задачи.**

Написать программу, которая производит манипуляции с числовыми матрицами, такие как:

1. Переставляя строки числовой матрицы размера M x N, упорядочить ее по убыванию элементов первого столбца.
2. Определить, является ли заданная целочисленная квадратная матрица порядка N магическим квадратом.
3. Определить, является ли заданная целочисленная квадратная матрица порядка N ортонормированной, т.е. такой, в которой скалярное произведение каждой пары различных строк равно 0, а скалярное произведение каждой строки на себя равно 1.
4. Дана действительная квадратная матрица порядка 2N. Получить новую матрицу, переставляя ее блоки размера N x N.
5. Найти определитель матрицы.

**Выполнение работы.**

Разобьем нашу программу на несколько функцию и блоков, для облегчения понимания кода программы:

1. GetMatrixSize(&column, &row) — вычисляем размерность матрицы из файла, методом подсчета количества строк и максимального количества пробелов в троке.
2. В функции main() создать двумерные динамические массивы для хранения числовых матриц.
3. LoadMatrixFromFile(matrixDefault) — загрузить матрицу из файла.
4. ShowMatrix(matrixDefault, column, row) — выводит исходную матрицу на экран.
5. SortMatrix(matrixNew,column,row) — сортирует матрицу и выводит на экран с помощью функции ShowMatrix(matrixDefault, column, row).
6. IsMagic(matrixDefault,column,row) — проверяет, является ли матрица магическим квадратом и выводит на экран результат.
7. IsOrtoNorm(matrixDefault, column, row) — проверят, является ли матрица ортонормированной и выводит результат.
8. Transposition(matrixTrans, column, row ) — переставляет блоки в матрице и выводит на экран с помощью функции ShowMatrix(matrixDefault, column, row).
9. В функции main() удалить все двумерные динамические массивы.

Исходный код программы представлен в приложении А.

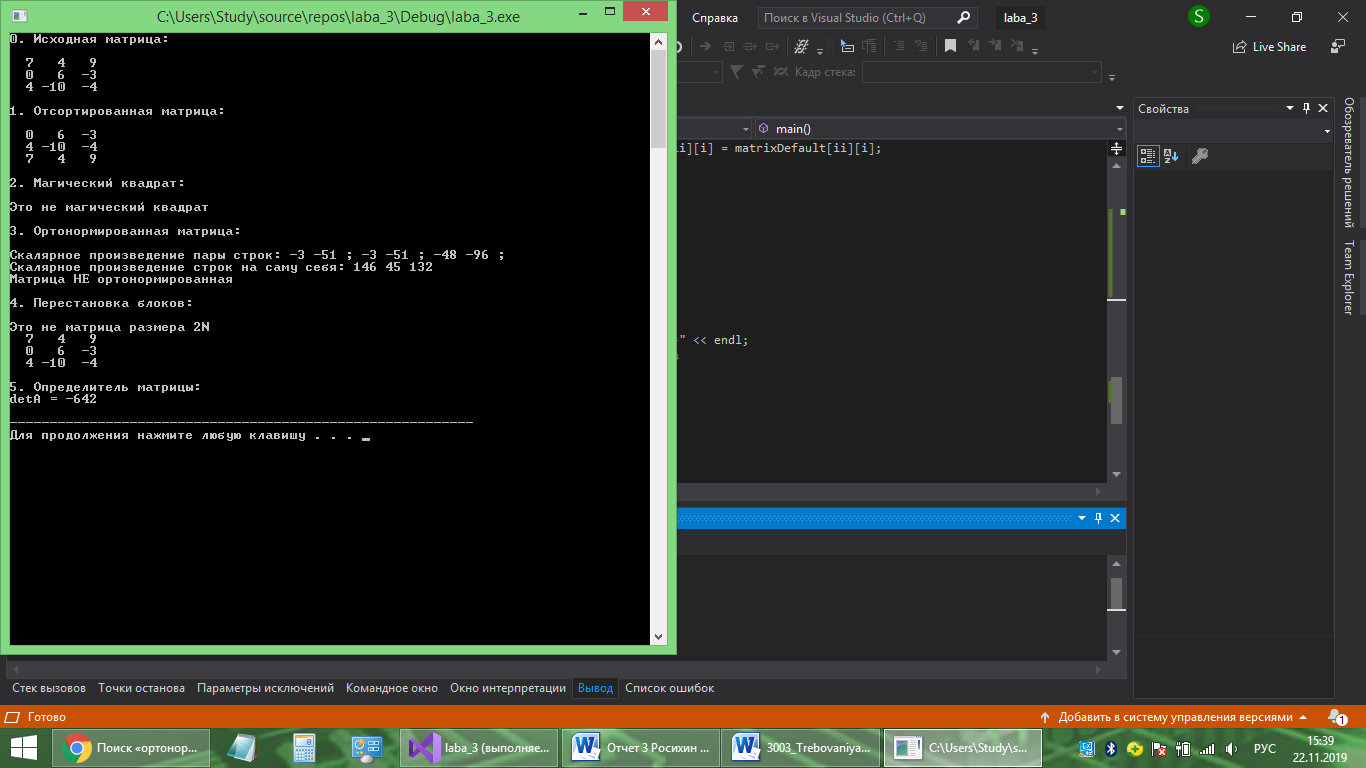
Скриншот работы экрана см. рис. 3.

Рисунок 3 — Скриншот работы программы.

**Выводы.**

Получение значений числовой матрицы удобнее всего реализовывать с помощью чтения данных из файла, чтобы каждый раз не вводить каждое значение с клавиатуры.

Сортировать числовую матрицу по столбцу удобно с помощью двух вложенных циклов и проверки соседнего значения. Перестановка блоков числовой матрицы происходит также при помощи вложенных циклов.

Вычисление определителя числовой матрицы происходи с помощью рекурсивной функции, которая вызывает сама себя, пока размер матрицы не уменьшится до 2.

Так как исходная матрица может быть и не квадратного вида, надо не забывать проверять, является она квадратной и отсекать те вычисления, которые должны производиться только с квадратными матрицами.

Для улучшения читаемости исходного кода, целесообразно разбивать всю программу на логические блоки и вставлять их в функции, в которые в свою очередь передаются либо переменные либо указатели на массивы.

Приложение А

**Исходный код программы.**

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

using namespace std;

void GetMatrixSize(int\* col, int\* row)

{

\*col = 0;

\*row = 0;

string s, ss;

ifstream file("matrix.txt");

int x = 0;

do

{

getline(file, s);

int y = 0;

int p = 0;

do

{

p = s.find(" ");

s.erase(0, p + 1);

y++;

} while (p > 0);

if (\*col < y) \*col = y;

x++;

} while (!file.eof());

\*row = x;

}

void LoadMatrixFromFile(int \*\*m)

{

string s, ss;

ifstream file("matrix.txt");

if (file)

{

int x = 0;

do

{

getline(file, s);

int y = 0;

int p = 0;

do

{

p = s.find(" ");

ss = s.substr(0, p);

s.erase(0, p + 1);

m[x][y] = atoi(ss.c\_str());

y++;

} while (p > 0);

x++;

} while (!file.eof());

} else cout << "Ошибка открытия файла.";

}

void SortMatrix(int\*\* m, int c, int r)

{

int temp[1000];

for (int i = 0; i < r - 1; i++)

for (int j = 0; j < r - i - 1; j++)

if (m[j][0] > m[j + 1][0])

for (int t=0; t<c ;t++)

{

temp[t] = m[j][t];

m[j][t] = m[j + 1][t];

m[j + 1][t] = temp[t];

}

}

void IsMagic(int\*\* m, int cc, int rr)

{

if (cc != rr) cout << "Матрица НЕ квадратная";

else

{

int etalon = 0, sum = 0;

bool magic = true;

int n = rr;

for (int i = 0; i < n; i++) etalon += m[0][i];

for (int ii = 1; ii < n; ii++) //строки

{

sum = 0;

for (int i = 0; i < n; i++) sum += m[ii][i];

if (etalon != sum)

{

magic = false;

break;

}

}

for (int ii = 0; ii < n; ii++) //столбцы

{

sum = 0;

for (int i = 0; i < n; i++) sum += m[i][ii];

if (etalon != sum)

{

magic = false;

break;

}

}

sum = 0;

for (int i = 0; i < n; i++) sum += m[i][i]; //главная диагональ

if (etalon != sum) magic = false;

sum = 0;

for (int i = 0; i < n; i++) sum += m[i][n - i - 1]; // побочная

if (etalon != sum) magic = false;

if (magic) cout << "Это магический квадрат. Магическая константа равна: " << etalon;

else cout << "Это не магический квадрат";

cout << endl << endl;

}

}

void IsOrtoNorm(int\*\* m, int c, int r)

{

int scalarMulti, scalarSelf;

bool isScalar = true;

cout << "Скалярное произведение пары строк: ";

for (int i3 = 0; i3 < r; i3++)

{

scalarMulti = 0;

for (int ii = 0; ii < r; ii++)

{

for (int i = 0; i < c; i++)

if (i3 != ii)scalarMulti += m[ii][i] \* m[i3][i];

if (scalarMulti != 0) isScalar = false;

if (i3 != ii) cout << scalarMulti << " ";

}

cout << "; ";

}

cout << endl;

cout << "Скалярное произведение строк на саму себя: ";

for (int ii = 0; ii < r; ii++)

{

scalarSelf = 0;

for (int i = 0; i < c; i++) scalarSelf += m[ii][i] \* m[ii][i];

if (scalarSelf != 1) isScalar = false;

cout << scalarSelf << " ";

}

if (isScalar) cout << endl << "Вывод. Матрица ортонормированная.";

else cout << endl <<"Матрица НЕ ортонормированная";

cout << endl << endl;

}

void Transposition(int\*\* m, int c, int r)

{

if (c == r && c % 2 == 0)

{

int n = c / 2;

int temp[100][100] = { 0 };

for (int ii = 0; ii < n; ii++) for (int i = 0; i < n; i++) temp[ii][i] = m[ii][i]; // temp

for (int ii = 0; ii < n; ii++) for (int i = 0; i < n; i++) m[ii][i] = m[ii + n][i]; //1=4

for (int ii = n; ii < n\*2; ii++) for (int i = 0; i < n; i++) m[ii][i] = m[ii][i + n]; //4=3

for (int ii = n; ii < n\*2; ii++) for (int i = n; i < n\*2; i++) m[ii][i] = m[ii - n][i]; //3=2

for (int ii = 0; ii < n; ii++) for (int i = n; i < n\*2; i++) m[ii][i] = temp[ii][i-n]; //2=temp

} else cout << "Это не матрица размера 2N" <<endl;

}

void ShowMatrix(int\*\* m, int c, int r)

{

for (int ii = 0; ii < r; ii++)

{

for (int i = 0; i < c; i++) cout << setw(3) << to\_string(m[ii][i]) << " ";

cout << endl;

}

cout << endl;

}

void GetMatr(int\*\* mas, int\*\* p, int i, int j, int m) {

// Получение матрицы без i-й строки и j-го столбца

int ki, kj, di, dj;

di = 0;

for (ki = 0; ki < m - 1; ki++) { // проверка индекса строки

if (ki == i) di = 1;

dj = 0;

for (kj = 0; kj < m - 1; kj++) { // проверка индекса столбца

if (kj == j) dj = 1;

p[ki][kj] = mas[ki + di][kj + dj];

}

}

}

int Determinant(int\*\* mas, int m)

{

// Рекурсивное вычисление определителя

int i, j, d, k, n;

int\*\* p;

p = new int\* [m];

for (i = 0; i < m; i++)

p[i] = new int[m];

j = 0; d = 0;

k = 1; //(-1) в степени i

n = m - 1;

if (m < 1) cout << "Определитель вычислить невозможно!";

if (m == 1) {

d = mas[0][0];

return(d);

}

if (m == 2) {

d = mas[0][0] \* mas[1][1] - (mas[1][0] \* mas[0][1]);

return(d);

}

if (m > 2) {

for (i = 0; i < m; i++) {

GetMatr(mas, p, i, 0, m);

d = d + k \* mas[i][0] \* Determinant(p, n);

k = -k;

}

}

return(d);

}

int main()

{

setlocale(0, "");

int column = 0, row=0;

GetMatrixSize(&column, &row);

int\*\* matrixDefault = new int\* [row]; // массив указателей

int\*\* matrixNew = new int\* [row];

int\*\* matrixTrans = new int\* [row];

for (int i = 0; i < row; i++) matrixDefault[i] = new int[column];

for (int i = 0; i < row; i++) matrixNew[i] = new int[column];

for (int i = 0; i < row; i++) matrixTrans[i] = new int[column];

LoadMatrixFromFile(matrixDefault);

cout << "0. Исходная матрица:" << endl << endl;

ShowMatrix(matrixDefault, column, row);

cout << "1. Отсортированная матрица:" << endl << endl;

for (int ii = 0; ii < row; ii++) for (int i = 0; i < column; i++) matrixNew[ii][i] = matrixDefault[ii][i];

SortMatrix(matrixNew,column,row);

ShowMatrix(matrixNew, column, row);

cout << "2. Магический квадрат:" << endl << endl;

IsMagic(matrixDefault,column,row);

cout << "3. Ортонормированная матрица: " << endl << endl;

IsOrtoNorm(matrixDefault, column, row);

cout << "4. Перестановка блоков: " << endl << endl;

for (int ii = 0; ii < row; ii++) for (int i = 0; i < column; i++) matrixTrans[ii][i] = matrixDefault[ii][i];

Transposition(matrixTrans, column, row );

ShowMatrix(matrixTrans, column, row);

cout << "5. Определитель матрицы: " << endl;

if (column == row)

{

int d = Determinant(matrixDefault, row);

cout << "detA = " << d << endl;

}

else cout << "Матрица не квадратная." << endl;

cout << endl << "----------------------------------------------------------" << endl;

for (int i = 0; i < row; i++) delete[] matrixDefault[i]; // удаляем массив

for (int i = 0; i < row; i++) delete[] matrixNew[i]; // удаляем массив

for (int i = 0; i < row; i++) delete[] matrixTrans[i]; // удаляем массив

system("pause");

}

**Список использованных источников.**

1. <http://cppstudio.com/post/446/>
2. <https://proginfo.ru/getline/>
3. https://ru.wikipedia.org/wiki/Определитель
4. https://ru.wikipedia.org/wiki/Магический\_квадрат