

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Информационных систем

ОТЧЕТ
по курсовой работе
по дисциплине «Алгоритмы и структуры данных»
ТЕМА: ЛИНЕЙНЫЕ СТРУКТУРЫ ДАННЫХ. РАЗРЕЖЕННЫЕ МАТРИЦЫ.

СТУДЕНТ ГР.0324

ПРЕПОДАВАТЕЛЬ

КОШЕЛЯЕВ А.С

МОЛДОВЯН Д.Н

САНКТ-ПЕТЕРБУРГ

2022

Цель работы.

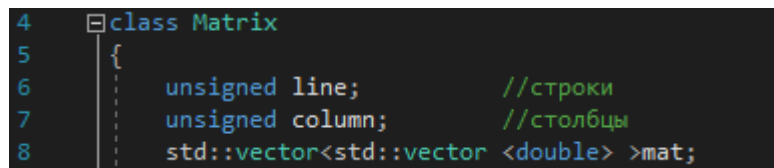
Разработать алгоритм и написать программу на языке C++, подсчёта и модификации согласно заданию, заданного блока матрицы.

Основные теоретические положения.

Матрица в математике — объект, записываемый в виде прямоугольной таблицы элементов кольца или поля, которая представляет собой совокупность строк и столбцов, на пересечении которых находятся её элементы.

Для данного проекта был написан класс матрица со всеми требуемыми в задании функциями. Класс является абстрактным типом данных, определяемым пользователем, и представляет собой модель реального объекта в виде данных и функций для работы с ними.

Краткое описание класса.

A screenshot of a code editor showing the definition of a C++ class named 'Matrix'. The code is as follows:

```
4 class Matrix
5 {
6     unsigned line;           //строки
7     unsigned column;        //столбцы
8     std::vector<std::vector<double>>> mat;
```

Рисунок 1 Поля класса.

Поля содержат количество строк столбцов и саму матрицу из вектора векторов с типом данных double. Так как строгой привязки в задании по типу данных не было, а из курса математики привычнее работать с цифрами был выбран такой формат. Хотя, несомненно, матрица могла содержать любой тип данных.

Методы класса.

Показаны на рис2.

```

unsigned get_line() const; //получить значение строк
unsigned get_column() const; //получить значение столбцов
Matrix(unsigned _l, unsigned _c, const double key); //параметрический конструктор аргументы; кол-во строк, кол-во столбцов (в м
double operator()(const unsigned& row, const unsigned& col) const; //доступ к элементам матрицы
Matrix operator=(const Matrix& rhs); //присваивания матриц
double sumElements() const; //сумма всех элементов
double multiElements() const; //произведение всех элементов
Matrix readfile(const char* name, unsigned _l, unsigned _c); //заполнение из файла; имя файла кол-во строк, кол-во столбцов
void PrintM(const Matrix& M); //вывод матрицы в консоль табличный вид
void MaxMin(const Matrix& M); //поиск max и min элемента матрицы
void MatrixLine(const Matrix& M); //последовательность элементов, полученную при обходе по строкам
void MatrixColumn(const Matrix& M); //последовательность элементов, полученную при обходе по столбцам
void LocalMaxMinMatrix(const Matrix& M); //список локальных максимумов минимумов
void MinLine(const Matrix& M); //минимум из максимальных элементов строк
void MaxColumn(const Matrix& M); //максимум из минимальных элементов столбцов
void SadPoint(const Matrix& M); //список седловых точек

```

Рисунок 2 Методы класса.

Самыми интересными были методы нахождения локальных максимумов и минимумов матрицы. В курсовом проекте применён метод обычного перебора всех элементов и сравнения их. Для этого метода уровень сложности $O(N \cdot N \cdot M \cdot M)$.

И метод поиска седловых точек. Седловой точкой называют элемент матрицы, такой, что он является наименьшим в своей строке и одновременно наибольшим в своем столбце, или, наоборот, наибольшим в своей строке и наименьшим в своем столбце.

- Найдём наименьшие значения по всем строкам, получим вектор A
- Найдём наибольшие значения по всем столбцам, получим вектор B
- Если максимальное значение вектора A меньше, чем минимальное значение вектора B, то седловых точек нет.
- Если максимальное значение вектора A равен минимальному значению вектора B, это значение S седловой точки.

Пример 1.

Задана матрица				наименьшее значение по строкам
8	7	0	6	0
6	8	5	10	5
8	8	5	10	максимальное значение по столбцам

т.к значение в строках равно значению в столбцах это и есть седловая точка с координатами (1,2) равная 5.

Постановка задачи.

Разработать программу подсчёта следующих характеристик заданного блока матрицы: ConsoleMatrix

1. Сумму и произведение всех элементов.
2. Максимальный и минимальный элементы.
3. Последовательность элементов, полученную при обходе по строкам (по столбцам).
- Интерфейс позволяющий задать какую строку или столбец вывести в консоль.
4. Список локальных максимумов (минимумов). Локальным максимумом называется элемент, не имеющий соседей больших (меньших), чем он сам. Соседями элемента являются элементы, ближайшие по вертикали, горизонтали или диагонали (если таковые имеются).

Подход: идея состоит в том, чтобы перебрать вектор векторов и проверить, является ли каждый элемент наименьшим или наибольшим среди смежных элементов. Если он наименьший, то это локальные минимумы, а если он наибольший, то это локальные максимумы. Ниже приведены шаги:

- Создайте два вектора `max` и `min` для хранения всех локальных максимумов и локальных минимумов.
- Матрица должна иметь размер не менее 2x2. (в противном случае не анализирую)
- Проверить углы матрицы. (там в соседях 3-и элемента)
- Проверить строчки верхнюю, нижнюю, и столбцы правый и левый. (там в соседях 5-ь элементов)
- Оставшиеся середину (там 8-м элементов)

5. Минимум из максимальных элементов строк.

- В каждой строке матрицы выбираю наибольшее значение (сколько строк столько значений). И выбираю из полученных значений минимальный.

6. Максимум из минимальных элементов столбцов.

- В каждом столбце матрицы выбираю наименьшее значение (сколько столбцов столько и значений). Из полученных значений выбрать максимальное.

7. Список седловых точек. Седловой точкой называют элемент матрицы, такой, что он является наименьшим в своей строке и одновременно наибольшим в своем столбце, или, наоборот, наибольшим в своей строке и наименьшим в своем столбце.

✓ Следует учесть, что если матрица имеет несколько седловых точек, то все их значения равны. Если все числа в матрице различны, то и седловой точки не более одной. Если все числа в матрице одинаковы, число седловых точек равно числу элементов.

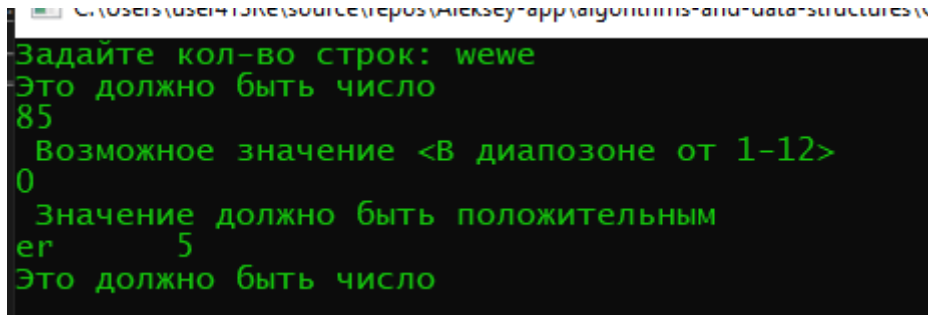
Выполнение работы.

Выполняя курсовую работу опираясь на полученные теоретические знания используя C++, получаем программу, выполняющую поставленные задачи.

```
#####@QQ@#####
ОСНОВНОЕ МЕНЮ ВВОД ДАННЫХ С КЛАВИАТУРЫ.
1.Создание матрицы. Задать кол-во строк и столбцов значение элементов матрицы выберется по умолчанию rand().
2.Создание матрицы из файла.
3.Сумма всех элементов.
4.Произведение всех элементов.
5.Максимальный и минимальный элементы.
6.Последовательность элементов, полученную при обходе по строкам
7.Последовательность элементов, полученную при обходе по столбцам
8.Список локальных максимумов (минимумов).
9.Минимум из максимальных элементов строк.
10.Максимум из минимальных элементов столбцов.
11.Список седловых точек.
12.Выход.
```

Рисунок 3. Меню программы.

Написанная программа содержит консольное меню из 12 пунктов позволяющая создавать и модифицировать матрицу согласно заданию. Также имеется проверка достоверности введенных значений.

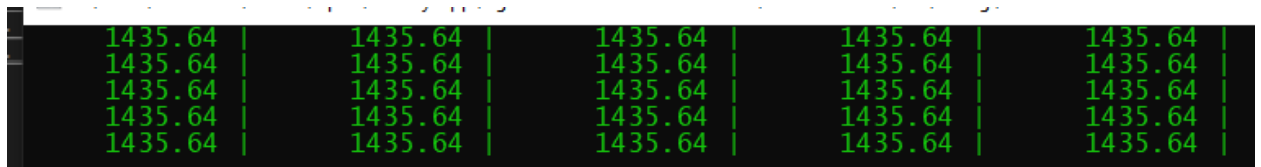


```

C:\Users\user4736\Documents\alexey-app\algorithms-and-data-structures\
Задайте кол-во строк: wewe
Это должно быть число
85
Возможное значение <В диапазоне от 1-12>
0
Значение должно быть положительным
er 5
Это должно быть число
  
```

Рисунок 4. Проверка введенных значений.

В первом пункте программа генерирует произвольное значение элемента, который в последствии заполняет матрицу заданного пользователем размера.

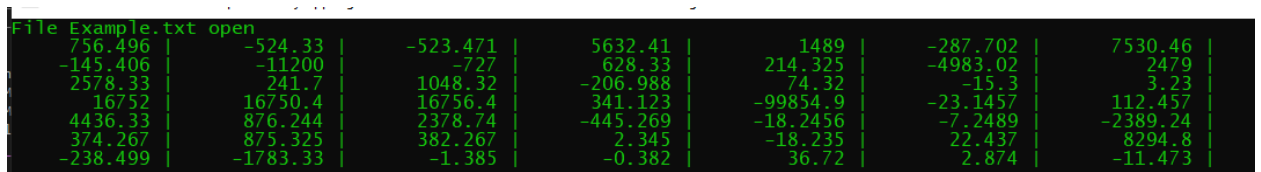


```

1435.64 | 1435.64 | 1435.64 | 1435.64 | 1435.64 |
1435.64 | 1435.64 | 1435.64 | 1435.64 | 1435.64 |
1435.64 | 1435.64 | 1435.64 | 1435.64 | 1435.64 |
1435.64 | 1435.64 | 1435.64 | 1435.64 | 1435.64 |
1435.64 | 1435.64 | 1435.64 | 1435.64 | 1435.64 |
  
```

Рисунок 5.

Создание матрица из файла пункт 2 по умолчанию формирует матрицу (7,7) из файла `Example.txt`



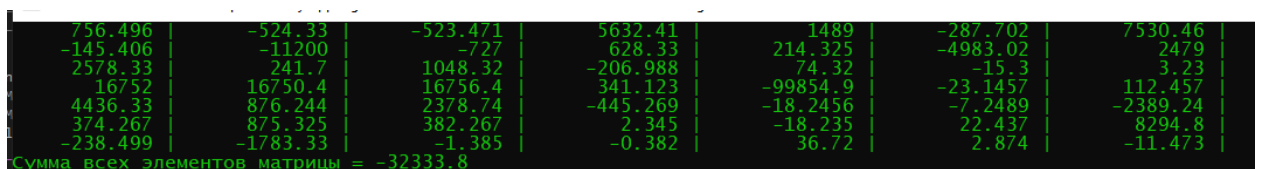
```

File Example.txt open
756.496 | -524.33 | -523.471 | 5632.41 | 1489 | -287.702 | 7530.46 |
-145.406 | -11200 | -727 | 628.33 | 214.325 | -4983.02 | 2479 |
2578.33 | 241.7 | 1048.32 | -206.988 | 74.32 | -15.3 | 3.23 |
16752 | 16750.4 | 16756.4 | 341.123 | -99854.9 | -23.1457 | 112.457 |
4436.33 | 876.244 | 2378.74 | -445.269 | -18.2456 | -7.2489 | -2389.24 |
374.267 | 875.325 | 382.267 | 2.345 | -18.235 | 22.437 | 8294.8 |
-238.499 | -1783.33 | -1.385 | -0.382 | 36.72 | 2.874 | -11.473 |
  
```

Рисунок 6.

Независимо от того, как в процессе выполнения программы сформирована матрица или вообще пропущены 1 и 2 программа позволяет выполнять оставшиеся 3-12 пункты т.к. в момент старта формируется матрица (2,2) и значением элементов равных 2.

Сумма всех элементов



```

756.496 | -524.33 | -523.471 | 5632.41 | 1489 | -287.702 | 7530.46 |
-145.406 | -11200 | -727 | 628.33 | 214.325 | -4983.02 | 2479 |
2578.33 | 241.7 | 1048.32 | -206.988 | 74.32 | -15.3 | 3.23 |
16752 | 16750.4 | 16756.4 | 341.123 | -99854.9 | -23.1457 | 112.457 |
4436.33 | 876.244 | 2378.74 | -445.269 | -18.2456 | -7.2489 | -2389.24 |
374.267 | 875.325 | 382.267 | 2.345 | -18.235 | 22.437 | 8294.8 |
-238.499 | -1783.33 | -1.385 | -0.382 | 36.72 | 2.874 | -11.473 |
Сумма всех элементов матрицы = -32333.8
  
```

Рисунок 7.

Произведение всех элементов

```
C:\Users\user413\source\repos\aleksey-app\algorithms-and-data-structures\ConsoleMatrix\X04\debug\ConsoleMatrix.exe
756.496 | -524.33 | -523.471 | 5632.41 | 1489 | -287.702 | 7530.46 |
-145.406 | -11200 | -727 | 628.33 | 214.325 | -4983.02 | 2479 |
2578.33 | 241.7 | 1048.32 | -206.988 | 74.32 | -15.3 | 3.23 |
16752 | 16750.4 | 16756.4 | 341.123 | -99854.9 | -23.1457 | 112.457 |
4436.33 | 876.244 | 2378.74 | -445.269 | -18.2456 | -7.2489 | -2389.24 |
374.267 | 875.325 | 382.267 | 2.345 | -18.235 | 22.437 | 8294.8 |
-238.499 | -1783.33 | -1.385 | -0.382 | 36.72 | 2.874 | -11.473 |
Произведение всех элементов матрицы = -5.4354e+122
```

Рисунок 8.

Максимальный и минимальные элементы.

```
756.496 | -524.33 | -523.471 | 5632.41 | 1489 | -287.702 | 7530.46 |
-145.406 | -11200 | -727 | 628.33 | 214.325 | -4983.02 | 2479 |
2578.33 | 241.7 | 1048.32 | -206.988 | 74.32 | -15.3 | 3.23 |
16752 | 16750.4 | 16756.4 | 341.123 | -99854.9 | -23.1457 | 112.457 |
4436.33 | 876.244 | 2378.74 | -445.269 | -18.2456 | -7.2489 | -2389.24 |
374.267 | 875.325 | 382.267 | 2.345 | -18.235 | 22.437 | 8294.8 |
-238.499 | -1783.33 | -1.385 | -0.382 | 36.72 | 2.874 | -11.473 |
Min element Matrix = -99854.9
Max element Matrix = 16756.4
```

Рисунок 9.

Пункт 6 последовательность элементов, полученную при обходе по строкам, имеет в своей реализации проверку вводимых данных.

```
756.496 | -524.33 | -523.471 | 5632.41 | 1489 | -287.702 | 7530.46 |
-145.406 | -11200 | -727 | 628.33 | 214.325 | -4983.02 | 2479 |
2578.33 | 241.7 | 1048.32 | -206.988 | 74.32 | -15.3 | 3.23 |
16752 | 16750.4 | 16756.4 | 341.123 | -99854.9 | -23.1457 | 112.457 |
4436.33 | 876.244 | 2378.74 | -445.269 | -18.2456 | -7.2489 | -2389.24 |
374.267 | 875.325 | 382.267 | 2.345 | -18.235 | 22.437 | 8294.8 |
-238.499 | -1783.33 | -1.385 | -0.382 | 36.72 | 2.874 | -11.473 |
Enter line number: 8
It should be the number 0
Enter line number: 0
Number 0 no
Enter line number: 8
Max number of line = 7
Enter line number: 5
Line 5
4436.33 | 876.244 | 2378.74 | -445.269 | -18.2456 | -7.2489 | -2389.24 |
```

Рисунок 10.

Функция содержит английские слова в соответствии с международной терминологией. Возможно будет применяться и в будущем поэтому не русифицировал.

Последовательность элементов, полученную при обходе по столбцам, также присутствует проверка вводимых значений.

```
756.496 | -524.33 | -523.471 | 5632.41 | 1489 | -287.702 | 7530.46 |
-145.406 | -11200 | -727 | 628.33 | 214.325 | -4983.02 | 2479 |
2578.33 | 241.7 | 1048.32 | -206.988 | 74.32 | -15.3 | 3.23 |
16752 | 16750.4 | 16756.4 | 341.123 | -99854.9 | -23.1457 | 112.457 |
4436.33 | 876.244 | 2378.74 | -445.269 | -18.2456 | -7.2489 | -2389.24 |
374.267 | 875.325 | 382.267 | 2.345 | -18.235 | 22.437 | 8294.8 |
-238.499 | -1783.33 | -1.385 | -0.382 | 36.72 | 2.874 | -11.473 |
Enter column number: 3
It should be the number 0
Enter column number: 0
Number 0 no
Enter column number: 15
Max number of column = 7
Enter column number: 7
Column 7
7530.46 |
2479 |
3.23 |
112.457 |
-2389.24 |
8294.8 |
-11.473 |
```

Рисунок 11.

Список локальных максимумов(минимумов)

```

756.496 | -524.33 | -523.471 | 5632.41 | 1489 | -287.702 | 7530.46 |
-145.406 | -11200 | -727 | 628.33 | 214.325 | -4983.02 | 2479 |
2578.33 | 241.7 | 1048.32 | -206.988 | 74.32 | -15.3 | 3.23 |
16752 | 16750.4 | 16756.4 | 341.123 | -99854.9 | -23.1457 | 112.457 |
4436.33 | 876.244 | 2378.74 | -445.269 | -18.2456 | -7.2489 | -2389.24 |
374.267 | 875.325 | 382.267 | 2.345 | -18.235 | 22.437 | 8294.8 |
-238.499 | -1783.33 | -1.385 | -0.382 | 36.72 | 2.874 | -11.473 |
Points of Local maxima found 8 : 756 7530 5632 112 8294 36 16752 16756
Points of Local minima found 6 : -11 -2389 -1783 -11200 -4983 -99854

```

Рисунок 12.

Минимум из максимальных элементов строк.

В каждой строке матрицы выбираю наибольшее значение (сколько строк столько значений). И выбираю из полученных значений минимальный.

```

756.496 | -524.33 | -523.471 | 5632.41 | 1489 | -287.702 | 7530.46 |
-145.406 | -11200 | -727 | 628.33 | 214.325 | -4983.02 | 2479 |
2578.33 | 241.7 | 1048.32 | -206.988 | 74.32 | -15.3 | 3.23 |
16752 | 16750.4 | 16756.4 | 341.123 | -99854.9 | -23.1457 | 112.457 |
4436.33 | 876.244 | 2378.74 | -445.269 | -18.2456 | -7.2489 | -2389.24 |
374.267 | 875.325 | 382.267 | 2.345 | -18.235 | 22.437 | 8294.8 |
-238.499 | -1783.33 | -1.385 | -0.382 | 36.72 | 2.874 | -11.473 |
Min 36.72

```

Рисунок 13.

Максимум из минимальных элементов столбцов. В каждом столбце матрицы выбираю наименьшее значение (сколько столбцов столько и значений). Из полученных значений выбрать максимальное.

```

756.496 | -524.33 | -523.471 | 5632.41 | 1489 | -287.702 | 7530.46 |
-145.406 | -11200 | -727 | 628.33 | 214.325 | -4983.02 | 2479 |
2578.33 | 241.7 | 1048.32 | -206.988 | 74.32 | -15.3 | 3.23 |
16752 | 16750.4 | 16756.4 | 341.123 | -99854.9 | -23.1457 | 112.457 |
4436.33 | 876.244 | 2378.74 | -445.269 | -18.2456 | -7.2489 | -2389.24 |
374.267 | 875.325 | 382.267 | 2.345 | -18.235 | 22.437 | 8294.8 |
-238.499 | -1783.33 | -1.385 | -0.382 | 36.72 | 2.874 | -11.473 |
Max -238.499

```

Рисунок 14.

Список седловых точек.

```

Список седловых точек.
Далеко не все матрицы имеют седловые точки.
Матрица сгенерированная в процессе работы алгоритма.
There are no saddle points in the matrix
Загружу подготовленный в процессе отладки пример с одной седловой точкой
Нажмите enter
File SadPointOne3x4.txt open
5 | 6 | 4 | 5 |
-2 | 5 | 3 | 7 |
8 | 7 | -2 | 6 |
All elements[-] 4 |
Загружу подготовленный в процессе отладки пример с несколькими точками
Нажмите enter
File Mat4x7.txt open
5.4321 | 5.4321 | 5.4321 | 5.4321 | 5.4321 | 5.4321 |
5 | 5 | 5 | 5 | 5 | 5 |
4 | 4 | 4 | 4 | 4 | 4 |
1 | 1 | 1 | 1 | 1 | 1 |
All elements[-] 5.4321 | 5.4321 | 5.4321 | 5.4321 | 5.4321 | 5.4321 |

```

Рисунок 15.

В курсовом проекте сформированы файлы `SadPointOne3x4.txt` и `Mat4x7.txt` для демонстрации нахождения седловых точек.

В процессе проверки можно сформировать свой текстовый файл и подключить к проекту для поиска седловых точек.

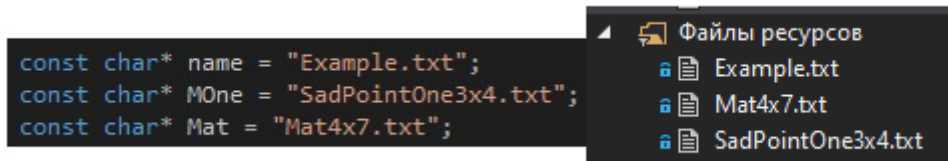


Рисунок 16.

Выводы.

Данная курсовая работа позволила объединить полученные знания, и позволила получить дополнительный опыт в написании кода C++.

При написании программы еще раз закрепил знания полученные в теории при изучении математики. При написании метода поиска седловых точек познакомился с Гессиан функциями описывающая поведение функции во втором порядке, применив которую можно найти седловую точку.

ПРИЛОЖЕНИЕ А

ПОЛНЫЙ КОД ПРОГРАММЫ

```
// ConsoleMatrix.cpp: Линейные структуры данных. Разреженные матрицы.
#include <iostream>
#include <windows.h>          //для меню
#include "Matrix.h"
using std::endl;
using std::cout;
using std::cin;
int check() {
    int qwe;
    while (true)
    {
        cin >> qwe;
        if (cin.fail())
        {
            cout << "Это должно быть число" << endl;
            cin.clear();
            cin.ignore(32767, '\n');
            continue;
        }
        if (qwe <= 0)
        {
            cout << "Значение должно быть положительным" << endl;
            cin.clear();
            cin.ignore(32767, '\n');
            continue;
        }
    }
}
```

```

    }
    if (qwe > 12)
    {
        cout << " Возможное значение <В диапазоне от 1-12>" << endl;
        cin.clear();
        cin.ignore(32767, '\n');
        continue;
    }
    else break;
}
system("cls");
cin.clear();
cin.ignore(32767, '\n');
return qwe;
}
int menu() {
    int one = 0;
    HANDLE 0 = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(0, FOREGROUND_RED | FOREGROUND_INTENSITY);
    cout << endl << "#####*@@@*#####" << endl;
    cout << "ОСНОВНОЕ МЕНЮ ВВОД ДАННЫХ С КЛАВИАТУРЫ." << endl;
    SetConsoleTextAttribute(0, FOREGROUND_GREEN | FOREGROUND_INTENSITY);
    cout << "1.Создание матрицы. Задать кол-во строк и столбцов значение элементов
матрицы выберется по умолчанию rand()." << endl;
    cout << "2.Создание матрицы из файла." << endl;
    cout << "3.Сумма всех элементов." << endl;
    cout << "4.Произведение всех элементов." << endl;
    cout << "5.Максимальный и минимальный элементы." << endl;
    cout << "6.Последовательность элементов, полученную при обходе по строкам" <<
endl;
    cout << "7.Последовательность элементов, полученную при обходе по столбцам" <<
endl;
    cout << "8.Список локальных максимумов (минимумов)." << endl;
    cout << "9.Минимум из максимальных элементов строк." << endl;
    cout << "10.Максимум из минимальных элементов столбцов." << endl;
    cout << "11.Список седловых точек." << endl;
    cout << "12.Выход." << endl;
    one = check();
    return one;
}
int main()
{
    const char* name = "Example.txt";
    const char* MOne = "SadPointOne3x4.txt";
    const char* Mat = "Mat4x7.txt";
    Matrix emptil(2, 2, 2);
    setlocale(LC_ALL, "Russian");
Mem1:
    int q = menu();
    if (q == 1) {
        cout << "Задайте кол-во строк: ";
        int l = check();
        cout << endl;
        cout << "Задайте кол-во столбцов: ";
        int c = check();
        double f = (double)rand() / RAND_MAX;
        double sum = f * (2547.34 - 0.00001);
        Matrix one(l, c, sum);
        emptil = one;
        one.PrintM(emptil);
        goto Mem1;
    }
    if (q == 2) {
        Matrix two = emptil.readfile(name, 7, 7);
    }
}

```

```

        emptil = two;
        two.PrintM(emptil);
        goto Mem1;
    }
    if (q == 3) {
        emptil.PrintM(emptil);
        cout << "Сумма всех элементов матрицы = ";
        double s = emptil.sumElements();
        cout << s << endl;
        goto Mem1;
    }
    if (q == 4) {
        emptil.PrintM(emptil);
        cout << "Произведение всех элементов матрицы = ";
        double m = emptil.multiElements();
        cout << m << endl;
        goto Mem1;
    }
    if (q == 5) {
        emptil.PrintM(emptil);
        emptil.MaxMin(emptil);
        goto Mem1;
    }
    if (q == 6) {
        emptil.PrintM(emptil);
        emptil.MatrixLine(emptil);
        goto Mem1;
    }
    if (q == 7) {
        emptil.PrintM(emptil);
        emptil.MatrixColumn(emptil);
        goto Mem1;
    }
    if (q == 8) {
        emptil.PrintM(emptil);
        emptil.LocalMaxMinMatrix(emptil);
        goto Mem1;
    }
    if (q == 9) {
        emptil.PrintM(emptil);
        emptil.MinLine(emptil);
        goto Mem1;
    }
    if (q == 10) {
        emptil.PrintM(emptil);
        emptil.MaxColumn(emptil);
        goto Mem1;
    }
    if (q == 11) {
        cout << "Список седловых точек." << endl;
        cout << "Далеко не все матрицы имеют седловые точки." << endl;
        cout << "Матрица сгенерированная в процессе работы алгоритма." << endl;
        emptil.SadPoint(emptil);
        cout << "Загружу подготовленный в процессе отладки пример с одной седловой
точкой" << endl;
        cout << "Нажмите enter";
        getchar();
        Matrix Eleven = emptil.readfile(MOne, 3, 4);
        emptil = Eleven;
        emptil.PrintM(emptil);
        emptil.SadPoint(emptil);
        cout << "Загружу подготовленный в процессе отладки пример с несколькими
точками" << endl;
        cout << "Нажмите enter";
    }
}

```

```

        getchar();
        Matrix ok = emptil.readfile(Mat, 4, 7);
        emptil = ok;
        emptil.PrintM(emptil);
        emptil.SadPoint(emptil);
        goto Mem1;
    }
    if (q == 12) {
        exit(EXIT_FAILURE);
    }
    return 0;
}

```

Matrix.h

```

//Определение класса матрица
#pragma once
#include <vector>
class Matrix
{
    unsigned line;           //строки
    unsigned column;         //столбцы
    std::vector<std::vector<double> >mat;
public:
    unsigned get_line() const;           //получить значение строк
    unsigned get_column() const;         //получить значение столбцов
    Matrix(unsigned _l, unsigned _c, const double key); //параметрический
    конструктор аргументы; кол-во строк, кол-во столбцов (в матрице) и значение элементов в
    матрице
    double operator()(const unsigned& row, const unsigned& col) const; //доступ к
    элементам матрицы
    Matrix operator=(const Matrix& rhs); //присваивания матриц
    double sumElements() const;         //сумма всех элементов
    double multiElements() const;       //произведение всех элементов
    Matrix readfile(const char* name, unsigned _l, unsigned _c); //заполнение из
    файла; имя файла кол-во строк, кол-во столбцов
    void PrintM(const Matrix& M);         //вывод матрицы в консоль табличный вид
    void MaxMin(const Matrix& M);         //поиск max и min элемента матрицы
    void MatrixLine(const Matrix& M); //последовательность элементов, полученную при
    обходе по строкам
    void MatrixColumn(const Matrix& M);   //последовательность элементов, полученную
    при обходе по столбцам
    void LocalMaxMinMatrix(const Matrix& M); //список локальных максимумов минимумов
    void MinLine(const Matrix& M);         //минимум из максимальных элементов строк
    void MaxColumn(const Matrix& M);       //максимум из минимальных элементов столбцов
    void SadPoint(const Matrix& M);        //список седловых точек
};

```

Matrix.cpp

```

//Релизация класса матрица
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include <vector>
#include "Matrix.h"
using std::cout;
using std::endl;
using std::ifstream;
using std::cin;
using std::vector;
unsigned Matrix::get_line() const
{

```

```

        return this->line;
    }
    unsigned Matrix::get_column() const
    {
        return this->column;
    }
    Matrix::Matrix(unsigned _l, unsigned _c, const double key)
    {
        mat.resize(_l);
        for (int i = 0; i < mat.size(); i++) {
            mat[i].resize(_c, key); //метод resize может принимать
необязательный аргумент, который инициализирует все элементы этим (key) значением
        }
        line = _l;
        column = _c;
    }
    double Matrix::operator()(const unsigned & row, const unsigned & col) const
    {
        return this->mat[row][col];
    }
    Matrix Matrix::operator=(const Matrix & rhs)
    {
        if (&rhs == this) {
            return *this;
        }
        unsigned new_rows = rhs.get_line();
        unsigned new_cols = rhs.get_column();
        mat.resize(new_rows);
        for (unsigned i = 0; i < mat.size(); i++) {
            mat[i].resize(new_cols);
        }
        for (unsigned i = 0; i < new_rows; i++) {
            for (unsigned j = 0; j < new_cols; j++) {
                mat[i][j] = rhs(i, j);
            }
        }
        line = new_rows;
        column = new_cols;
        return *this;
    }
    double Matrix::sumElements() const
    {
        double sum = 0;
        for (int i = 0; i < line; i++) {
            for (int j = 0; j < column; j++) {
                sum += this->mat[i][j];
            }
        }
        return sum;
    }
    double Matrix::multiElements() const
    {
        double multiplication = 1;
        for (int i = 0; i < line; i++) {
            for (int j = 0; j < column; j++) {
                multiplication *= this->mat[i][j];
            }
        }
        return multiplication;
    }
    Matrix Matrix::readfile(const char * name, unsigned _l, unsigned _c)
    {
        ifstream qmatrix(name);
        if (!qmatrix) {

```

```

        cout << "File " << name << " error!!!" << endl;
        exit(EXIT_FAILURE);
    }
    else {
        cout << "File " << name << " open" << endl;
    }
    double key = 0.0;
    Matrix rfile(_l, _c, 0.0);
    for (unsigned i = 0; i < _l; i++) {
        for (unsigned j = 0; j < _c; j++) {
            qmatrix >> key;
            rfile.mat[i][j] = key;
        }
    }
    qmatrix.close();
    return rfile;
}

void Matrix::PrintM(const Matrix & M)
{
    for (int i = 0; i < M.line; i++) {
        for (int j = 0; j < M.column; j++) {
            cout << std::setw(12) << M(i, j) << " | ";
        }
        cout << endl;
    }
}

void Matrix::MaxMin(const Matrix & M)
{
    double min = M.mat[0][0], max = M.mat[0][0];
    for (int i = 0; i < M.line; i++) {
        for (int j = 0; j < M.column; j++) {
            if (min > M.mat[i][j]) {
                min = M.mat[i][j];
            }
            if (max < M.mat[i][j]) {
                max = M.mat[i][j];
            }
        }
    }
    cout << "Min element Matrix = " << min << endl;
    cout << "Max element Matrix = " << max << endl;
}

void Matrix::MatrixLine(const Matrix & M)
{
    unsigned key;
    while (true) {
        cout << "Enter line number ";
        cin >> key;
        if (cin.fail()) {
            cout << "It should be the number" << endl;
            cin.clear();
            cin.ignore(32767, '\n');
            continue;
        }
        if (key == 0) {
            cout << "Number 0 no" << endl;
            continue;
        }
        if (key > M.line) {
            cout << "Max number of line = " << M.line << endl;
            continue;
        }
        else break;
    }
}

```

```

        /*system("cls");*/
        cout << "Line " << key << endl;
        for (int j = 0; j < M.column; j++) {
            cout << std::setw(12) << M(key-1, j) << " | ";
        }
        cout << endl;
    }
}
void Matrix::MatrixColumn(const Matrix & M)
{
    unsigned key;
    while (true) {
        cout << "Enter column number      ";
        cin >> key;
        if (cin.fail()) {
            cout << "It should be the number" << endl;
            cin.clear();
            cin.ignore(32767, '\n');
            continue;
        }
        if (key == 0) {
            cout << "Number 0 no" << endl;
            continue;
        }
        if (key > M.column) {
            cout << "Max number of column = " << M.column << endl;
            continue;
        }
        else break;
    }
    /*system("cls");*/
    cout << "Column " << key << endl;
    for (int j = 0; j < M.line; j++) {
        cout << std::setw(12) << M(j, key - 1) << " | ";
        cout << endl;
    }
}
void Matrix::LocalMaxMinMatrix(const Matrix & M) {
    vector<double> max, min;    //empty vector to store points of local maxima and
    minima
    int MaxSum = 0, MinSum = 0;
    if (M.line >= 2 && M.column >= 2) {
        if (M.mat[0][0] > M.mat[0][1] && M.mat[0][0] > M.mat[1][1] && M.mat[0][0] >
M.mat[1][0]) {    //левый верхний угол
            max.push_back(M.mat[0][0]);
            MaxSum++;
        }
        else if (M.mat[0][0] < M.mat[0][1] && M.mat[0][0] < M.mat[1][1] &&
M.mat[0][0] < M.mat[1][0]) {
            min.push_back(M.mat[0][0]);
            MinSum++;
        }
        //правый верхний угол
        if (M.mat[0][column - 1] > M.mat[0][column - 2] && M.mat[0][column - 1] >
M.mat[1][column - 2] && M.mat[0][column - 1] > M.mat[1][column - 1]) {
            max.push_back(M.mat[0][column-1]);
            MaxSum++;
        }
        else if (M.mat[0][column - 1] < M.mat[0][column - 2] && M.mat[0][column -
1] < M.mat[1][column - 2] && M.mat[0][column - 1] < M.mat[1][column - 1]) {
            min.push_back(M.mat[0][column-1]);
            MinSum++;
        }
        //правый нижний угол
    }
}

```

```

        if (M.mat[line - 1][column - 1] > M.mat[line - 1][column - 2] && M.mat[line
- 1][column - 1] > M.mat[line - 2][column - 2] && M.mat[line - 1][column - 1] >
M.mat[line - 2][column - 1]) {
            max.push_back(M.mat[line-1][column-1]);
            MaxSum++;
        }
        else if (M.mat[line - 1][column - 1] < M.mat[line - 1][column - 2] &&
M.mat[line - 1][column - 1] < M.mat[line - 2][column - 2] && M.mat[line - 1][column - 1]
< M.mat[line - 2][column - 1]) {
            min.push_back(M.mat[line-1][column-1]);
            MinSum++;
        }
        //левый нижний угол
        if (M.mat[line - 1][0] > M.mat[line - 2][0] && M.mat[line - 1][0] >
M.mat[line - 2][1] && M.mat[line - 1][0] > M.mat[line - 1][1]) {
            max.push_back(M.mat[line-1][0]);
            MaxSum++;
        }
        else if (M.mat[line - 1][0] < M.mat[line - 2][0] && M.mat[line - 1][0] <
M.mat[line - 2][1] && M.mat[line - 1][0] < M.mat[line - 1][1]) {
            min.push_back(M.mat[line-1][0]);
            MinSum++;
        }
        //верхняя строка
        if (M.column > 2) {
            for (int j = 1; j < M.column - 1; j++) {
                if (M.mat[0][j] > M.mat[0][j - 1] && M.mat[0][j] > M.mat[1][j
- 1] && M.mat[0][j] > M.mat[1][j] && M.mat[0][j] > M.mat[1][j + 1] && M.mat[0][j] >
M.mat[0][j + 1]) {
                    max.push_back(M.mat[0][j]);
                    MaxSum++;
                }
                if (M.mat[0][j] < M.mat[0][j - 1] && M.mat[0][j] < M.mat[1][j
- 1] && M.mat[0][j] < M.mat[1][j] && M.mat[0][j] < M.mat[1][j + 1] && M.mat[0][j] <
M.mat[0][j + 1]) {
                    min.push_back(M.mat[0][j]);
                    MinSum++;
                }
            }
        }
        //правая боковая колонка
        if (M.line > 2) {
            for (int i = 1; i < M.line - 1; i++) {
                if (M.mat[i][column - 1] > M.mat[i - 1][column - 1] &&
M.mat[i][column - 1] > M.mat[i - 1][column - 2] && M.mat[i][column - 1] > M.mat[i][column
- 2] && M.mat[i][column - 1] > M.mat[i + 1][column - 2] && M.mat[i][column - 1] > M.mat[i
+ 1][column - 1]) {
                    max.push_back(M.mat[i][column-1]);
                    MaxSum++;
                }
                if (M.mat[i][column - 1] < M.mat[i - 1][column - 1] &&
M.mat[i][column - 1] < M.mat[i - 1][column - 2] && M.mat[i][column - 1] < M.mat[i][column
- 2] && M.mat[i][column - 1] < M.mat[i + 1][column - 2] && M.mat[i][column - 1] < M.mat[i
+ 1][column - 1]) {
                    min.push_back(M.mat[i][column-1]);
                    MinSum++;
                }
            }
        }
        //нижняя строка матрицы
        if (M.column > 2) {
            for (int j = 1; j < M.column - 1; j++) {

```



```

        if (M.mat[line - 1][j] > M.mat[line - 1][j - 1] && M.mat[line
- 1][j] > M.mat[line - 2][j - 1] && M.mat[line - 1][j] > M.mat[line - 2][j] && M.mat[line
- 1][j] > M.mat[line - 2][j + 1] && M.mat[line - 1][j] > M.mat[line - 1][j + 1]) {
            max.push_back(M.mat[line-1][j]);
            MaxSum++;
        }
        if (M.mat[line - 1][j] < M.mat[line - 1][j - 1] && M.mat[line
- 1][j] < M.mat[line - 2][j - 1] && M.mat[line - 1][j] < M.mat[line - 2][j] && M.mat[line
- 1][j] < M.mat[line - 2][j + 1] && M.mat[line - 1][j] < M.mat[line - 1][j + 1]) {
            min.push_back(M.mat[line-1][j]);
            MinSum++;
        }
    }
}
//левая боковая колонка
if (M.line > 2) {
    for (int i = 1; i < M.line - 1; i++) {
        if (M.mat[i][0] > M.mat[i - 1][0] && M.mat[i][0] > M.mat[i -
1][1] && M.mat[i][0] > M.mat[i][1] && M.mat[i][0] > M.mat[i + 1][1] && M.mat[i][0] >
M.mat[i + 1][0]) {
            max.push_back(M.mat[i][0]);
            MaxSum++;
        }
        if (M.mat[i][0] < M.mat[i - 1][0] && M.mat[i][0] < M.mat[i -
1][1] && M.mat[i][0] < M.mat[i][1] && M.mat[i][0] < M.mat[i + 1][1] && M.mat[i][0] <
M.mat[i + 1][0]) {
            min.push_back(M.mat[i][0]);
            MinSum++;
        }
    }
}
//середина матрицы
if (M.line >= 3 && M.column >= 3) {
    for (int i = 1; i < M.line-1; i++) {
        for (int j = 1; j < M.column-1; j++) {
            if (M.mat[i][j] > M.mat[i][j - 1] && M.mat[i][j] >
M.mat[i - 1][j - 1] && M.mat[i][j] > M.mat[i - 1][j] && M.mat[i][j] > M.mat[i - 1][j + 1]
&& M.mat[i][j] > M.mat[i][j + 1]
&& M.mat[i][j] > M.mat[i + 1][j + 1] &&
M.mat[i][j] > M.mat[i + 1][j] && M.mat[i][j] > M.mat[i + 1][j - 1]) {
                max.push_back(M.mat[i][j]);
                MaxSum++;
            }
            if (M.mat[i][j] < M.mat[i][j - 1] && M.mat[i][j] <
M.mat[i - 1][j - 1] && M.mat[i][j] < M.mat[i - 1][j] && M.mat[i][j] < M.mat[i - 1][j + 1]
&& M.mat[i][j] < M.mat[i][j + 1]
&& M.mat[i][j] < M.mat[i + 1][j + 1] &&
M.mat[i][j] < M.mat[i + 1][j] && M.mat[i][j] < M.mat[i + 1][j - 1]) {
                min.push_back(M.mat[i][j]);
                MinSum++;
            }
        }
    }
}
}
if (max.size() > 0) {
    cout << "Points of Local maxima found " << MaxSum << " : ";
    for (int a : max) {
        cout << a << " ";
    }
    cout << endl;
}
else {
    cout << "There are no points of local max" << endl;
}
}

```

```

        if (min.size() > 0) {
            cout << "Points of Local minima found " << MinSum << " : ";
            for (int a : min) {
                cout << a << " ";
            }
            cout << endl;
        }
        else {
            cout << "There are no points of local min" << endl;
        }
    }
    else {
        cout << "Data analysis requires a matrix, at least 2x2" << endl;
        exit(EXIT_FAILURE);
    }
}

void Matrix::MinLine(const Matrix & M)
{
    vector<double> sum;
    for (int i = 0; i < M.line; i++) {
        double max = M.mat[i][0];
        for (int j = 0; j < M.column; j++) {
            if (max < M.mat[i][j]) {
                max = M.mat[i][j];
            }
        }
        sum.push_back(max);
    }
    std::sort(sum.begin(), sum.end());
    cout << "Min " << sum[0] << endl;
}

void Matrix::MaxColumn(const Matrix & M)
{
    vector<double> sum;
    int one = 0;
    for (int i = 0; i < M.column; i++) {
        double min = M.mat[0][i];
        for (int j = 0; j < M.line; j++) {
            if (min > M.mat[j][i]) {
                min = M.mat[j][i];
            }
        }
        sum.push_back(min);
        one++;
    }
    std::sort(sum.begin(), sum.end());
    cout << "Max " << sum[one-1] << endl;
}

void Matrix::SadPoint(const Matrix & M)
{
    vector<double> Aline, Bcolumn, Happens;
    unsigned sum = 0;
    for (int i = 0; i < M.line; i++) {
        double min = M.mat[i][0];
        for (int j = 0; j < M.column; j++) {
            if (min > M.mat[i][j]) {
                min = M.mat[i][j];
            }
            else if (min == M.mat[i][j]) {
                Happens.push_back(min);
                sum++;
            }
        }
        Aline.push_back(min);
    }
}

```

```

    }
    for (int i = 0; i < M.column; i++) {
        double max = M.mat[0][i];
        for (int j = 0; j < M.line; j++) {
            if (max < M.mat[j][i]) {
                max = M.mat[j][i];
            }
            else if (max == M.mat[j][i]) {
                Happens.push_back(max);
                sum++;
            }
        }
        Bcolumn.push_back(max);
    }
    std::sort(Aline.begin(), Aline.end());
    std::sort(Bcolumn.begin(), Bcolumn.end());
    if (Aline[Aline.size() - 1] < Bcolumn[0]) {
        cout << "There are no saddle points in the matrix" << endl;
    }
    else if (sum / 2 == M.column*M.line) {
        cout << "All elements" << sum / 2 << endl;
        cout << "All elements of the matrix are equal " << Happens[0] << endl;
    }
    else if (Aline[Aline.size() - 1] == Bcolumn[0] && sum / 2 != M.column*M.line) {
        vector<double> number;
        for (double sumA : Aline) {
            for (double sumB : Bcolumn) {
                if (sumA == sumB) {
                    number.push_back(sumB);
                }
            }
        }
        cout << "All elements[-] ";
        for (double n : number) {
            cout << n << " | ";
        }
        cout << endl;
    }
}
}

```

Дополнительные файлы и полный код разработки совместно с пояснительной запиской находится.

[Github.com](https://github.com)