

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет компьютерных наук
Департамент программной инженерии**

**НАХОЖДЕНИЕ РЕШЕНИЯ СИСТЕМЫ ЛИНЕЙНЫХ УРАВНЕНИЙ С
ПОМОЩЬЮ ФОРМУЛ КРАМЕРА (вариант 8)**

Отчёт

Дисциплина: «Архитектура вычислительных систем»

Исполнитель:
студент группы БПИ198
Гудзикович М. С.

Москва 2020

СОДЕРЖАНИЕ

1. ТЕКСТ ЗАДАНИЯ	3
2. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ	3
3. ТЕСТИРОВАНИЕ ПРОГРАММЫ	4
ИСТОЧНИКИ	5
ПРИЛОЖЕНИЕ 1	6
КОД ПРОГРАММЫ	6

1. ТЕКСТ ЗАДАНИЯ

Используя формулы Крамера, найти решение системы линейных уравнений.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = b_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = b_4 \end{cases}$$

Предусмотреть возможность деления на ноль. Входные данные: коэффициенты системы. Оптимальное количество потоков выбрать самостоятельно.

2. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ

Программа запускается из консоли, обязательный входной параметр – путь до файла, содержащего коэффициенты системы.

Формат строки с путём до файла test: path-to\test (например, C:\Users\max_g\Desktop\ABC\test1.txt).

Чтобы программа работала корректно – необходимо соблюдать формат входного файла. Формат – четыре строки, в каждой четыре действительных числа через пробел. Во избежание непредвиденного поведения не рекомендуется вводить большие числа (большие 10000 по модулю).

Программа работает по парадигме взаимодействующих данных [2]. Такая модель была выбрана в связи с тем, что в данной задаче мы имеем фиксированное количество подзадач – мы должны посчитать 5 различных определителей квадратных матриц размера 4. Логично распараллелить задачу, чтобы не тратить много времени на последовательное вычисление определителей.

Программа работает следующим образом: она считывает данные из файла, преобразовывая их в матрицу (для этого специально написан класс Matrix). Затем, мы запускаем пять потоков – i-ый для вычисления определителя матрицы СЛАУ без столбца i. Определитель – статическая функция, считается рекурсивно методом разложения по первой строке. Далее проверяется определитель матрицы A на равенство 0, после чего в консоль выводится решение системы с помощью правила Крамера (или программа сообщает о вырожденности матрицы A).

3. ТЕСТИРОВАНИЕ ПРОГРАММЫ

(C) корпорация Майкрософт (Microsoft Corporation), 2019. Все права защищены.

```
C:\Users\maxma\Desktop\Ass\task03>task03 C:\Users\maxma\Desktop\Ass\task03\tests\test1.txt
Input coeffs:
1      2      3      4      5
6      7      8      6      4
1      2      2      7      1
3      2      4      5      5

x1 = -1.85366
x2 = -2.09756
x3 = 3.78049
x4 = -0.0731707

C:\Users\maxma\Desktop\Ass\task03>
```

Рисунок 1 – Обычный пример, на который программа даёт верный ответ

```
C:\Users\maxma\Desktop\Ass\task03>task03 C:\Users\maxma\Desktop\Ass\task03\tests\test2.txt
Input coeffs:
1      2      0      3      1
3      2      0      0      2
2      0      0      1     -4
0      0      0     -1     -4

Det(A) = 0! Kramer's method is useless here (the system has 0 or +oo number of solutions)
```

Рисунок 2 – Пример, в котором определитель матрицы равен нулю.

```
C:\Users\maxma\Desktop\Ass\task03>task03 C:\Users\maxma\Desktop\Ass\task03\tests\test3.txt
Input coeffs:
10.1  12.2  3.09  -7.15  4
121.01 -2    3     43.2  1
1      3     3     2     1
-10   20    5     6     1

x1 = 0.0720713
x2 = 0.0478991
x3 = 0.397491
x4 = -0.204121
```

Рисунок 3 – Пример, в котором есть отрицательные и нецелые числа.

(Все ответы совпадают с реальными)

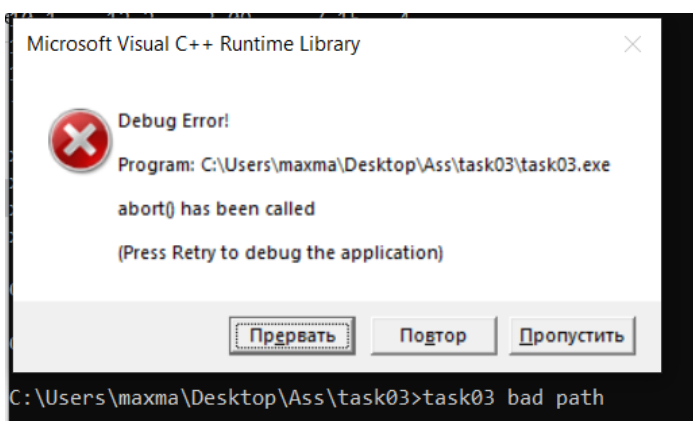


Рисунок 4 - Также если ввести не один входной параметр, а несколько (или если по указанному пути не будет файла) – программа выдаст исключение (почему-то без сообщения).

ИСТОЧНИКИ

1. SoftCraft, сайт по учебной дисциплине. [Электронный ресурс] <http://softcraft.ru/> (дата обращения: 7.11.2020).
2. Парадигмы параллельного программирования, Блог Программиста. [Электронный ресурс] <https://pro-prof.com/forums/topic/parallel-programming-paradigms> (дата обращения: 9.11.2020).

КОД ПРОГРАММЫ

```

#include <iostream>
#include <fstream>
#include <vector>
#include <thread>

using namespace std;

class Matrix // Класс матриц
{
private:
    int n, m; // n columns and m rows
    vector<double> data;

public:
    Matrix(int cols, int rows)
    {
        n = cols;
        m = rows;
        data = vector<double>(n * m);
    }

    double& operator()(int i, int j)
    {
        return data[i * n + j];
    }

    double operator()(int i, int j) const
    {
        return data[i * n + j];
    }

    void print() const
    {
        for (int i = 0; i < m; ++i) {
            for (int j = 0; j < n; ++j)
                cout << this->operator()(i, j) << "\t";
            cout << endl;
        }
    }

    bool isSquare() const { return n == m && n > 0; }
    int getN() const { return n; }
    int getM() const { return m; }

    Matrix excludeCol(int q) const // returns original matrix without column q
    {

```

```

Matrix temp(getN() - 1, getM());
int i = 0, j = 0; // here we save indexes of elements in temp

if (q == -1)
    q = n - 1;

for (int row = 0; row < m; ++row)
{
    for (int col = 0; col < n; ++col)
    {
        if (col != q)
        {
            temp(i, j) = this->operator()(row, col);
            j++;

            if (j == n - 1)
            {
                j = 0;
                i++;
            }
        }
    }
}
return temp;
}

```

```

Matrix cofactor(int p, int q) const
{
    Matrix temp(getN() - 1, getM() - 1);
    int i = 0, j = 0; // here we save indexes of elements in temp

    for (int row = 0; row < m; ++row)
    {
        for (int col = 0; col < n; ++col)
        {
            if (row != p && col != q)
            {
                temp(i, j) = this->operator()(row, col);
                j++;

                if (j == n - 1)
                {
                    j = 0;
                    i++;
                }
            }
        }
    }
    return temp;
}

```

```

    }

    static double det(const Matrix& m)
    {
        if (!m.isSquare())
            throw new exception("can't calculate a det of non-square matrix");

        double D = 0;
        int n = m.getN();

        if (n == 1)
            return m(0, 0);

        Matrix temp(n - 1, n - 1);
        int sign = 1;

        for (int j = 0; j < n; j++)
        {
            temp = m.cofactor(0, j);
            D += sign * m(0, j) * Matrix::det(temp);
            sign = -sign;
        }

        return D;
    }
};

void detWrapper(const Matrix& m, double& det)
{
    det = Matrix::det(m);
}

void sqsum(int iTread, int iTN, double* arr, int size, double& sum) {
    for (int i = iTread; i < size; i += iTN) {
        /*sum += arr[i] * arr[i];
        sum += arr[i] * arr[i];
    }
}

void solve(Matrix& m)
{
    thread* thr[5];
    double x[5];
    double sum[5];

    for (int i = 0; i < 5; ++i)
    {
        Matrix temp = m.excludeCol(i);

```



```

        //temp.print();
        //cout << endl;
        //thr[i] = new std::thread{ sqsum, i, 5, x, 5, std::ref(sum[i]) };
        thr[i] = new thread{ detWrapper, temp, ref(x[i])};
    }

    for (int i = 0; i < 5; ++i)
    {
        thr[i]->join();
        delete thr[i];
    }

    if (x[4] == 0)
        cout << "Det(A) = 0! Kramer's method is useless here (the system has 0 or +oo
number of solutions)" << endl;
    else
    {
        cout << "x1 = " << -x[0] / x[4] << endl;
        cout << "x2 = " << x[1] / x[4] << endl;
        cout << "x3 = " << -x[2] / x[4] << endl;
        cout << "x4 = " << x[3] / x[4] << endl;
    }

    return;
}

int main(int argc, char** argv)
{
    //int argc = 2;
    //string argv[] = { "asd", "C:\\Users\\maxma\\Desktop\\Ass\\task03\\tests\\test1.txt" };

    if (argc != 2) {
        throw new invalid_argument("Only one parameter - path to test - expected");
    }
    //cout << "Input format - four strings, each contains coeffs\n For example, for {i} string it
is 'ai1 ai2 ai3 ai4 bi'\n";

    ifstream in(argv[1]);
    Matrix m(5, 4);

    for (int i = 0; i < 4; ++i)
        for (int j = 0; j < 5; ++j)
            in >> m(i, j);

    cout << "Input coeffs:\n";
    m.print();
    cout << endl;
    //cout << "---\n";

```

```
//m.excludeCol(-1).print();  
  
solve(m);  
}
```