**Национальный исследовательский университет**

**«Высшая школа экономики»**

# Факультет компьютерных наук

Департамент

# Программной инженерии

**Домашняя работа по дисциплине**

**«Архитектура вычислительных систем»**

Тема работы: Вариант 8. Используя формулы Крамера, найти решение системы линейных уравнений.

Выполнил: студент группы БПИ194

Гребенщиков М. М.

тел. +7 (922) 704 5875

e-mail адрес: [mmgrebenschikov@edu.hse.ru](mailto:mmgrebenschikov@edu.hse.ru) Преподаватель: Легалов Александр Иванович

Москва, 2020

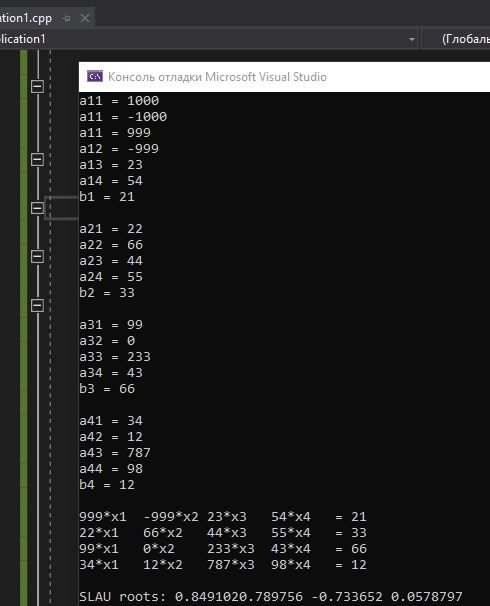
Структура работы

1. Kramer.cpp – содержит исходный код программы
2. inputData – тестовые наборы
3. Note.pdf – Отчёт

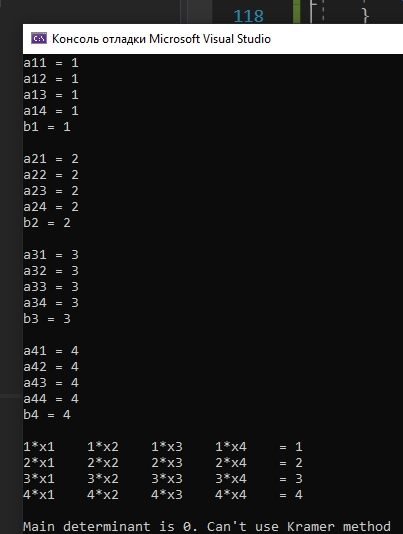
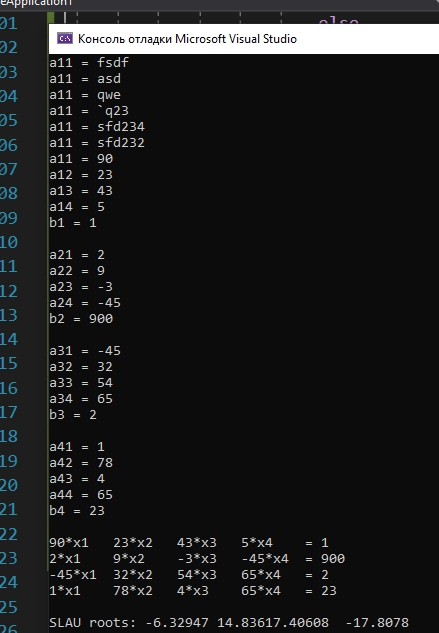
В коде программы находятся комментарии, описывавшие каждый шаг программы.

## Тестирование программы

В качестве входных данных программа принимает коэффициенты СЛАУ из консоли. Пользователь вводит по 1 числу. Область допустимых значений: [-999; 999]. Предусмотрена обработка некорректных данных.



Test1



Test2 Test3

## Алгоритм работы программы

1. Считывание коэффициентов СЛАУ
2. Вычисление основного определителя
3. Проверка определителя на равенство 0
4. Запуск 4 поток вычисления корней
5. Каждый поток выводит подсчитанный xi на экран

## Список литературы

1. https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4\_%D0%9A%D1%80%D0%B0%D0%BC%D0%B5%D1%80%D0%B0

## Приложение. Код программы

#include <iostream>

#include <fstream>

#include <thread>

#include <stdlib.h>

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <string>

//Метод вывода СЛАУ на экран

void printSLAU(int arc[4][5]) {

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 5; j++) {

if (j == 4)

std::cout << "= " << arc[i][j] << std::endl;

else

std::cout << arc[i][j] << "\*x" << j + 1 << "\t";

}

}

}

//Вычисление определителя 3 на 3

long long calcDetThreeXThree(int arc[3][3]) {

long long det = arc[0][0] \* arc[1][1] \* arc[2][2] + arc[0][1] \* arc[1][2] \* arc[2][0] + arc[1][0] \* arc[2][1] \* arc[0][2];

det -= (arc[0][2] \* arc[1][1] \* arc[2][0] + arc[1][0] \* arc[0][1] \* arc[2][2] + arc[0][0] \* arc[2][1] \* arc[1][2]);

return det;

}

//Вычисление определителя 4 на 4

long long calcDetFourXFour(int arc[4][5]) {

long long det = 0;

int minor[3][3];

//Разложение матрицы по первой строке

for (int i = 0; i < 4; i++) {

for (int j = 1; j < 4; j++)

{

int ind = 0;

//Составление минора

for (int k = 0; k < 4; k++)

{

if (k == i)

continue;

minor[j - 1][ind] = arc[j][k];

ind++;

}

}

//Вычисление определителя минора

long long minorDet = calcDetThreeXThree(minor);

if (i % 2 == 1)

minorDet \*= -1;

det += (long long)arc[0][i] \* minorDet;

}

return det;

}

void calcRoot(int rootNum, int arc[4][5], long long mainDet) {

//Замена стобца коэфициентов свободными членами

int arrc[4][5];

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++) {

if (j == rootNum)

{

arrc[i][j] = arc[i][4];

}

else

{

arrc[i][j] = arc[i][j];

}

}

}

//Вычисление определителя

long long det = calcDetFourXFour(arrc);

//Вывод и вычисление корня

std::cout << (double)det / (double)mainDet << " ";

}

int main()

{

int arc[4][5];

//Считывание входных данных из консоли

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 5; j++)

{

while (true)

{

try

{

std::string s;

if (j != 4)

std::cout << "a" << i + 1 << j + 1 << " = ";

else

std::cout << "b" << i + 1 << " = ";

std::cin >> s;

arc[i][j] = std::stoi(s);

//Обрабаботка некорректных входных данных

if (arc[i][j] >= 1000 || arc[i][j] <= -1000)

continue;

break;

}

catch (std::invalid\_argument) {

continue;

}

}

}

std::cout << std::endl;

}

std::ifstream in("intput/");

//Вывод СЛАУ на экран

printSLAU(arc);

//Высиление детерминанта коэфициентов

long long mainDet = calcDetFourXFour(arc);

//Если детерминант 0 - Крамера использовать нельзя

if (mainDet == 0)

{

std::cout << std::endl << "Main determinant is 0. Can't use Kramer method";

return 0;

}

std::cout << std::endl << "SLAU roots: ";

//Запуск поток для вычисления корней

std::thread tA(calcRoot, 0, arc, mainDet);

std::thread tB(calcRoot, 1, arc, mainDet);

std::thread tC(calcRoot, 2, arc, mainDet);

std::thread tD(calcRoot, 3, arc, mainDet);

tA.join();

tB.join();

tC.join();

tD.join();

}