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

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

по дисциплине «Программирование»

Тема: Использование массивов для решения геометрических задач

Студент гр. 2302 _____ Коняев А.Е.

Преподаватель _____ Калмычков В.А.

Санкт-Петербург
2022

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Коняев А.Е.

Группа 2302

Тема работы:

Использование массивов для решения геометрических задач

Исходные данные:

Файл с входными данными координат точек и количеством точек, через которые строятся прямые и находятся пары прямых с наибольшим углом.

Содержание пояснительной записки:

“Введение”, “Исходная формулировка задания”, “Анализ задания и выполнение контрольного примера”, “Математическая постановка задачи” (“Исходные данные”, “Ограничения на исходные данные”, “Результирующие данные”, “Связь выходных данных с исходными данными”), “Особенности решения задания на компьютере”, “Форматы представления данных на внешних носителях при использовании файлов”, “Внутренний формат представления данных”, “Разбиение основной задачи на подзадачи”, “Описание метода решения отдельных подзадач”, “Оформление описания алгоритма в виде блок-схемы”, “Текст программы”, “Тестовые примеры”, “Выводы”.

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 17.11.2022

Дата сдачи реферата:

Дата защиты реферата:

Студент _____ Коняев А.Е.

Преподаватель _____ Калмычков В.А.

АННОТАЦИЯ

В пояснительной записке к курсовой работе подробно описан принцип работы программы, решающей поставленную задачу: описание каждой функции и их параметры, тип и назначение каждой переменной, анализ задания, блок-схемы. Для нахождения наибольшего угла используются функции из библиотеки math.h, для вывода данных в файл - переменные типа fstream из библиотеки fstream, а для организации вывода данных - строковый тип из библиотеки string. В последних главах приведён непосредственно текст программы и разобраны различные возможные ситуации, с которыми программа успешно справляется.

SUMMARY

The explanatory note to the course work describes in detail the principle of operation of the program that solves the task: a description of each function and their parameters, the type and purpose of each variable, task analysis, flowcharts. To find the largest angle, functions from the math.h library are used, variables of the fstream type from the fstream library are used to output data to a file, and a string type from the string library is used to organize data output. In the last chapters, the text of the program is given directly and various possible situations with which the program successfully copes are analyzed.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель:

Написать программу, которая будет решать поставленную задачу с использованием двухмерного массива.

Задачи:

- Придумать метод нахождения наибольшего угла
- Написать функцию для нахождения коэффициентов уравнения прямых
- Написать функцию для нахождения реального количества точек
- Написать функцию ввода значений в массив
- Написать функцию обработки значений и побочные функции, используемые в ней
- Сформировать программу из функций
- Описать метод решения задания, функции и переменные
- Сделать блок-схемы каждой функции

1. ИСХОДНАЯ ФОРМУЛИРОВКА ЗАДАНИЯ

Выбрать из всех прямых, проходящих через любую пару из заданных N точек на плоскости, проходящие условию: пары прямых, образующие между собой больший угол.

2. АНАЛИЗ ЗАДАНИЯ И ВЫПОЛНЕНИЕ КОНТРОЛЬНОГО ПРИМЕРА

Для решения задачи будем использовать метод линейной интерполяции для нахождения функции по двум заданным точкам. После нахождения функций двух прямых можно использовать коэффициент k уравнения $y = kx + b$, так как он является тангенсом угла наклона нашей прямой к положительному направлению оси Ox. Далее, использовав функцию арктангенса, мы найдем два угла наклона прямых относительно оси Ox и с их помощью сможем вычислить угол между прямыми. При решении задачи работа идёт преимущественно с коэффициентами уравнения $y = kx + b$, но для записи уравнений используется каноническое уравнение прямой $Ax + By + C = 0$, так как с его помощью можно правильно записать прямые параллельные оси Oy. Также предлагается сначала перебрать все прямые для нахождения наибольшего угла, а потом найти пары прямых с таким углом. Отметим, что углом между двумя прямыми считается острый угол или прямой, если они перпендикулярны, то есть $0° \leq \alpha \leq 90°$.

Для контрольного примера возьмём простой набор из четырёх точек, его несложно проверить вручную. Возьмём точки $(0, 0), (-10, 0), (-10, -10), (0, -10)$, они являются вершинами квадрата, а значит наибольший угол между прямыми будет равен $90°$. Пар прямых с таким углом должно быть 5, 4 пары из прямых, образующих стороны квадрата, и пересечение диагоналей.

```
4
0 0
-10 -10
0 -10
-10 0
```

Рисунок 1. Контрольный прмер во входном файле in.txt

Построим все эти прямые и обозначим углы.

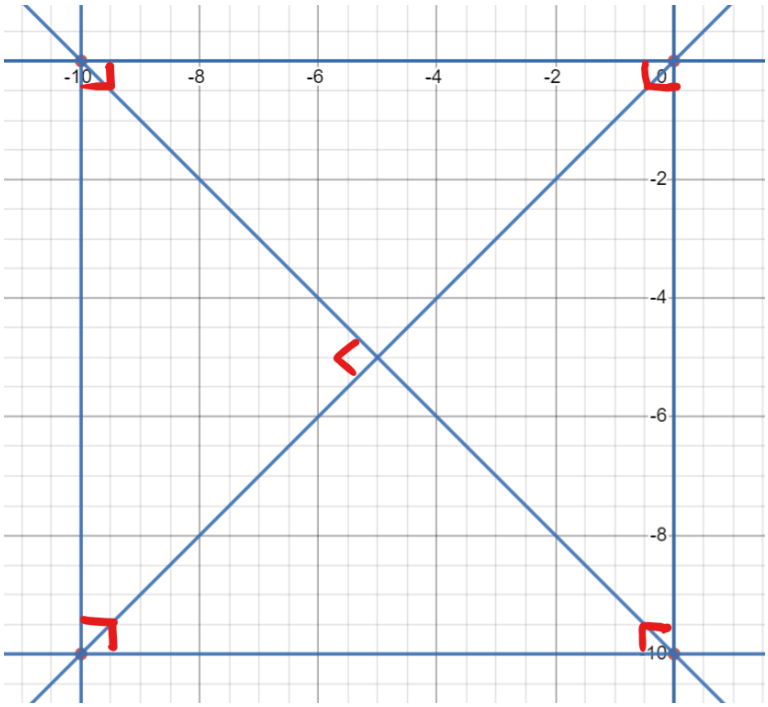

Рисунок 2. График в desmos с обозначенными искомыми углами

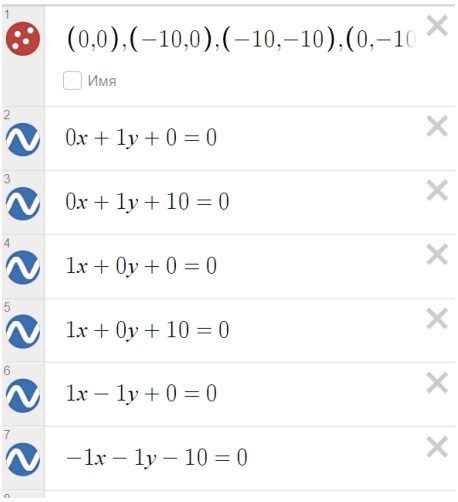

Рисунок 3. Уравнения прямых в desmos

Как видно на рисунке 2 таких прямых действительно оказалось 5. Теперь проверим контрольный пример на программе и сравним с результатами, полученными вручную.

```
Первая прямая: 1.00x-1.00y+0.00=0                Вторая прямая: -1.00x-1.00y-10.00=0              Угол: 90
(0, 0), (-10, -10)                               (0, -10), (-10, 0)

Первая прямая: 1.00x+0.00y-0.00=0                Вторая прямая: 0.00x-1.00y+0.00=0                Угол: 90
(0, 0), (0, -10)                                 (0, 0), (-10, 0)

Первая прямая: 1.00x+0.00y-0.00=0                Вторая прямая: 0.00x-1.00y-10.00=0               Угол: 90
(0, 0), (0, -10)                                 (-10, -10), (0, -10)

Первая прямая: 0.00x-1.00y+0.00=0                Вторая прямая: 1.00x+0.00y+10.00=0               Угол: 90
(0, 0), (-10, 0)                                 (-10, -10), (-10, 0)

Первая прямая: 0.00x-1.00y-10.00=0               Вторая прямая: 1.00x+0.00y+10.00=0               Угол: 90
(-10, -10), (0, -10)                             (-10, -10), (-10, 0)
```

Рисунок 4. Результат работы программы в файле out.txt

Данные полностью совпали, теперь посмотрим на файл протокола.

```
Данные корректны
Данные корректны
Данные корректны
Данные корректны

4
0    0
-10  -10
0    -10
-10  0

Первая прямая: 1.00x-1.00y+0.00=0                Вторая прямая: 1.00x-1.00y+0.00=0                Угол: 0
(0, 0), (-10, -10)                               (0, 0), (-10, -10)

Первая прямая: 1.00x-1.00y+0.00=0                Вторая прямая: 1.00x+0.00y-0.00=0                Угол: 45
(0, 0), (-10, -10)                               (0, 0), (0, -10)

Первая прямая: 1.00x-1.00y+0.00=0                Вторая прямая: 0.00x-1.00y+0.00=0                Угол: 45
(0, 0), (-10, -10)                               (0, 0), (-10, 0)

Первая прямая: 1.00x-1.00y+0.00=0                Вторая прямая: 0.00x-1.00y-10.00=0               Угол: 45
(0, 0), (-10, -10)                               (-10, -10), (0, -10)

Первая прямая: 1.00x-1.00y+0.00=0                Вторая прямая: 1.00x+0.00y+10.00=0               Угол: 45
(0, 0), (-10, -10)                               (-10, -10), (-10, 0)

Первая прямая: 1.00x-1.00y+0.00=0                Вторая прямая: -1.00x-1.00y-10.00=0              Угол: 90
(0, 0), (-10, -10)                               (0, -10), (-10, 0)

Первая прямая: 1.00x+0.00y-0.00=0                Вторая прямая: 1.00x+0.00y-0.00=0                Угол: 0
(0, 0), (0, -10)                                 (0, 0), (0, -10)

Первая прямая: 1.00x+0.00y-0.00=0                Вторая прямая: 0.00x-1.00y+0.00=0                Угол: 90
(0, 0), (0, -10)                                 (0, 0), (-10, 0)
```

Рисунок 5. Часть данных из файла протокола Protocol.txt

На рисунке 5 видно, что помимо нужных нам пар прямых с углом 90°, есть ещё пары с углами 0° и 45°, то есть параллельные прямые-стороны (либо проходящие через одинаковые точки) и пересечение сторон и диагоналей, что полностью соответствует рисунку 2.

3. МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ

3.1 ИСХОДНЫЕ ДАННЫЕ

На вход программе подаётся N точек с координатами x и y. Координаты имеют вещественный тип float.

```
4
0.23 12.234
-10.2 1
0 -1
-1.98 0
```

Рисунок 6. Пример входных данных

Для записи координат точек используется двухмерный массив.

3.2 ОГРАНИЧЕНИЯ НА ИСХОДНЫЕ ДАННЫЕ

Так как для записи координат x и y используется тип данных float[] они ограниченны диапазоном $[-3.4 \times 10^{38};\ 3.4 \times 10^{38}]$. Также размер двухмерного массива ограничен константным значением $N = 100$, то есть количество точек не может быть больше 100.

3.3 РЕЗУЛЬТИРУЮЩИЕ ДАННЫЕ

Пример результирующих данных представлен на рисунке 4. Они представляют собой пары канонических уравнений прямых с указанием точек, через которые они проведены, и угла между ними.

3.4 СВЯЗЬ ВЫХОДНЫХ ДАННЫХ С ИСХОДНЫМИ ДАННЫМИ

Коэффициенты A, B и C канонического уравнения $Ax + By + C = 0$, связаны с коэффициентами k и b уравнения $y = kx + b$ следующими соотношениями: $k = -\frac{A}{B}$, $b = -\frac{C}{B}$, а коэффициенты k и b в свою очередь связаны с исходными координатами точек $(x_1, y_1), (x_2, y_2)$ таким образом: $k = \frac{y_1 - y_2}{x_1 - x_2}$, $b = y_1 - x_1 * k$

4. ОСОБЕННОСТИ РЕШЕНИЯ ЗАДАНИЯ НА КОМПЬЮТЕРЕ

При решении задачи использовались переменные типа float, int, unsigned, const unsigned, bool, char, string и fstream. Отсюда, $A[i][j]$, max_$corn$, $corn$, $k1$, $k2$, $b1$, $b2 \in [-3.4 \times 10^{38}; 3.4 \times 10^{38}]$, a, i, $count\_ws$, $position$, k, $i1$, $i2$, $j1$, $j2$, $size$, $q \in [0; 4\,294\,967\,295]$, $num \in [-2\,147\,483\,647; 2\,147\,483\,647]$, $flag$, ws_flag, $break\_flag$, $string\_flag$ принимают значения 0 или 1.

5. ФОРМАТЫ ПРЕДСТАВЛЕНИЯ ДАННЫХ НА ВНЕШНИХ НОСИТЕЛЯХ ПРИ ИСПОЛЬЗОВАНИИ ФАЙЛОВ

in.txt:

N

d.ddd d.ddd

d.ddd d.ddd

...

Protocol.txt:

Макет O1:

“Данные корректны”(1) или “Количество чисел в строке больше 2”(2) или “Обнаружена пустая строка”(3) или “Недостаточно данных”(4)

…

Макет O2:

a

d.ddd d.ddd

d.ddd d.ddd

...

Макет O3:

Первая прямая: $Ax + By + C = 0$ Вторая прямая: $Ax + By + C = 0$

Угол: d

(x_1, y_1) (x_2, y_2)

…

out.txt:

Макет O1:

Первая прямая: $Ax + By + C = 0$ Вторая прямая: $Ax + By + C = 0$

Угол: d

(x_1, y_1) (x_2, y_2)

…

6. ВНУТРЕННИЙ ФОРМАТ ПРЕДСТАВЛЕНИЯ ДАННЫХ

Макет O1: Не удалось открыть входной файл

Макет O2: Не удалось открыть файл протокола

Макет O3: Не удалось открыть выходной файл

Тип	Название	Назначение
float[]	A[]	Двухмерный массив с числами из файла in.txt
int	num	Временная переменная для считывания лишних значений из файла
unsigned	i	Итерируемая переменная цикла
	q	
	i1	
	j1	
	i2	
	j2	
	a	Реальное количество строк
	position	Переменная для сохранения позиции начала строки
	size	Подсчёт непустых строк
	count_ws	Подсчёт пробелов
	m	Параметр столбца для задания
float	max_corn	Наибольший угол
	corn	Переменная для нахождения угла
	k1	Угловой коэффициент первой прямой
	k2	Угловой коэффициент второй прямой
	b1	Свободный коэффициент первой прямой
	b2	Свободный коэффициент второй прямой
const unsigned	n	Максимальное количество строк исходной матрицы
	pi	Число Пи
bool	flag	Флаг для проверки на пустую строку
	ws_flag	Флаг для проверки на подряд идущие пробелы
	break_flag	Флаг для выхода из бесконечного цикла при нахождении конца строки
	string_flag	Флаг для определения есть ли строка в массиве

char	tmp	Временная переменная для хранения текущего символа
fstream	f	Файл in.txt
	g	
	p	Файл Protocol.txt
	e	Файл out.txt
string	points1	Первая пара точек для вывода в файл
	points2	Вторая пара точек для вывода в файл
	tmp1	Первая комбинация из пар точек для записи в массив
	tmp2	Вторая комбинация из пар точек для записи в массив
	s1	Первая строка с уравнениями прямых
	s2	Вторая строка с уравнениями прямых
string[]	points_memor	Массив для запоминания предыдущих комбинаций из пар точек

7. РАЗБИЕНИЕ ОСНОВНОЙ ЗАДАЧИ НА ПОДЗАДАЧИ

Имя функции	Назначение	Параметры				Внешние эффекты
		входные	выходные	модифицируемые	транзитные	
Interpolation	Нахождение коэффициентов k и b по заданным точкам	x1, x2, y1, y2		k, b		
SizeArr	Определение размера массива			a, p		В файл протокола записывается информация о введённых данных
InpArr	Заполнение массива числами из in.txt	a		A[n][2], p		В файл протокола записываются введённые значения
PrintPoints	Печатает заданные точки в файл	A[n][2], i1, i2, j1, j2		p		В файл записываются точки
PointsMemor	Запоминание комбинаций из пар точек, через которые уже проводились прямые	A[n][2], i1, i2, j1, j2	1 или 0	B[10000], k		
Process	Обработка введённых значений и получение результата	A[n][2], a		p, e		В файл протокола записываются все прямые с углами между ними, а в выходной файл результат
main	Основная функция					

8. ОПИСАНИЕ МЕТОДА РЕШЕНИЯ ОТДЕЛЬНЫХ ПОДЗАДАЧ

В функции SizeArr для каждой строки находится количество чисел в строке относительно пробельных символов (с помощью флагов учитываются

пустые стоки, пробелы перед первым числом и подряд идущие пробелы), а реальный размер массива увеличивается только в том случае, когда в строке больше одного числа.

В функции InpArr в начале каждой строки происходит запоминание текущей позиции в файле с помощью метода tellg(), потом считываются символы без пропуска разделителей, определяется количество чисел в строке и, когда каретка доходит до символа перехода на следующую строку, если в строке достаточно символов, каретка переходит на начало строки и считывает 2 числа с пропуском разделителей в массив, а остальные пропускает в режиме без пропуска разделителей пока не дойдёт до перехода на следующую строку. Если же символов в строке недостаточно, переход на новую строку осуществляется сразу, без возвращения на изначальную позицию.

В функции Process сначала находится наибольший угол: перебираются все возможные комбинации пар прямых, с помощью функции Interpolation находятся коэффициенты k и b, потом через функцию atan(), вычитание одного угла из другого, взятие модуля функцией abs() и при необходимости вычитания этого значения из 180°, если угол оказался больше 90°, находится угол между прямыми, а затем сравнивается с текущим значением максимального угла, которое перезаписывается, если новый угол больше. Далее начинается новый перебор: сначала происходит проверка с помощью функции PointsMemor на то, что такой комбинации пар точек ещё не было, а потом угол находится так же, как и при нахождении наибольшего угла, после чего уравнение прямой, угол и точки записываются в файл протокола и, если угол оказался равен наибольшему, всё это записывается и в выходной файл.

9. ОФОРМЛЕНИЕ ОПИСАНИЯ АЛГОРИТМА В ВИДЕ БЛОК-СХЕМЫ

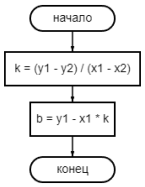

Рисунок 7. Блок-схема функции Interpolation

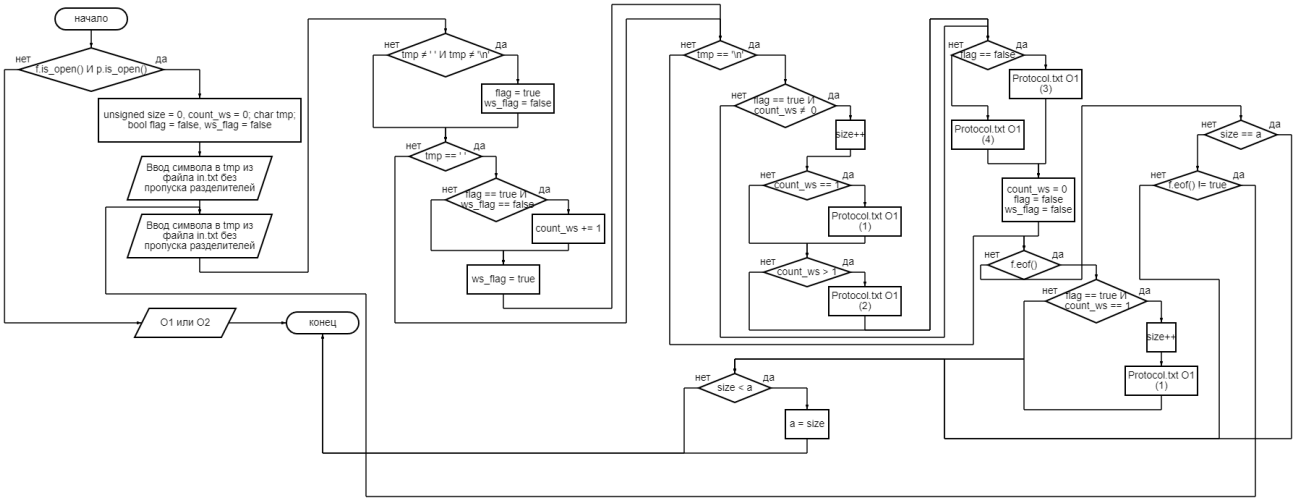

Рисунок 8. Блок-схема функции SizeArr

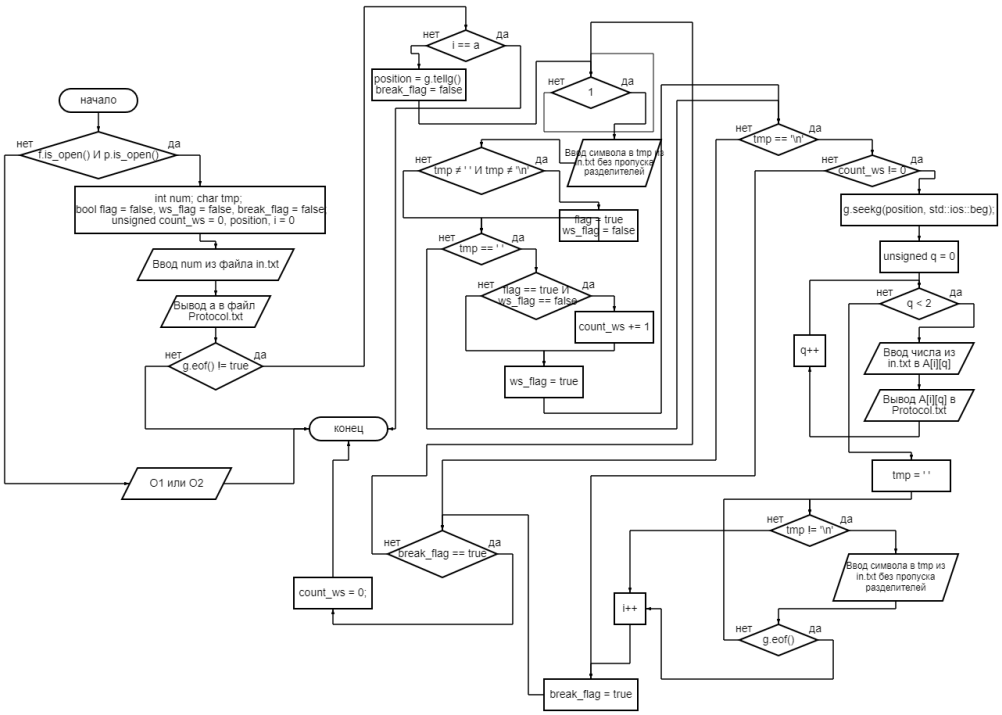

Рисунок 9. Блок-схема функции InpArr

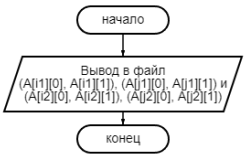

Рисунок 10. Блок-схема функции PrintPoints

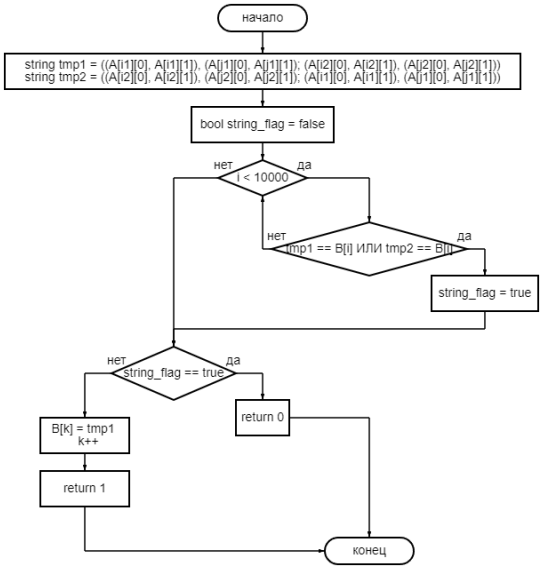

Рисунок 11. Блок-схема функции PointsMemor

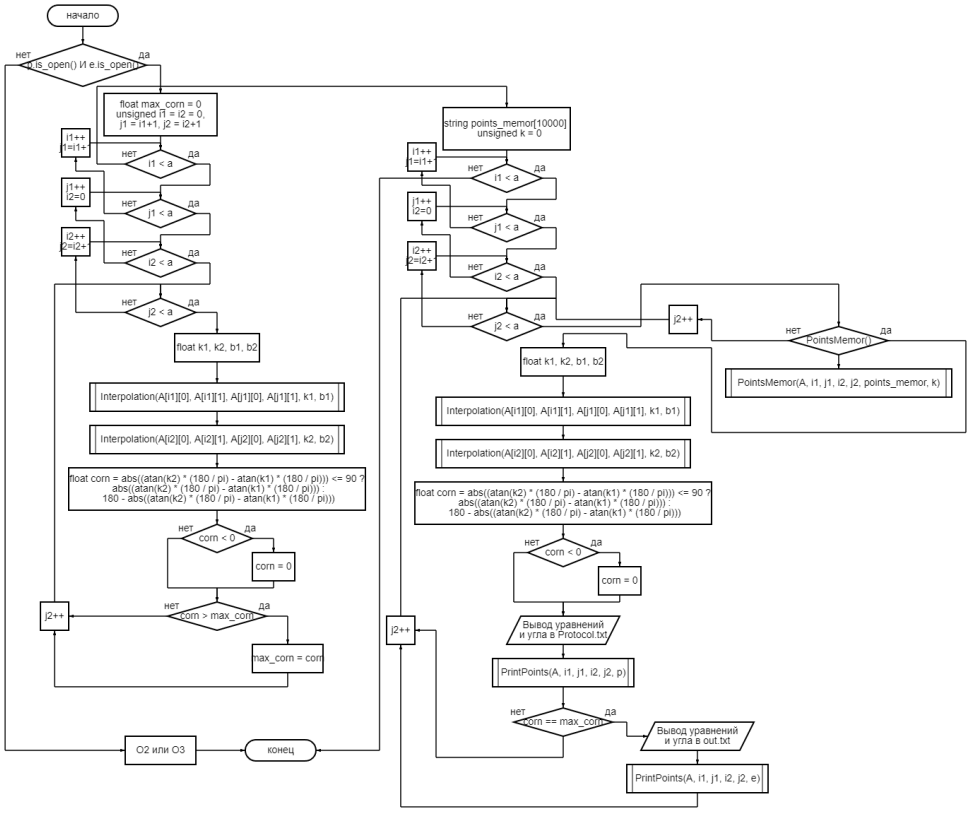

Рисунок 12. Блок-схема функции Process

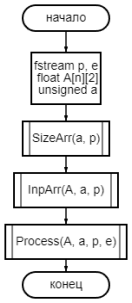

Рисунок 13. Блок-схема функции main

10. ТЕКСТ ПРОГРАММЫ

```
#include <iostream>
#include <iomanip>
#include <math.h>
#include <fstream>
#include <string>
#include <locale.h>

const int n = 100;
const double pi = 3.14159265358979323846;

void Interpolation(float x1, float y1, float x2, float y2, float& k, float& b) {
    k = (y1 - y2) / (x1 - x2);
    b = y1 - x1 * k;
}

void SizeArr(unsigned& a, std::fstream& p) {
      std::fstream f; f.open("in.txt", std::ios::in);

      if (f.is_open() && p.is_open()) {
            f >> a;
            unsigned size = 0;
            char tmp;
            if (a > n) a = n;
            unsigned count_ws = 0; bool flag = false, ws_flag = false;
            f >> std::noskipws >> tmp;

            do {
                  f >> std::noskipws >> tmp;
                  if (tmp != ' ' && tmp != '\n') {
                        flag = true;
                        ws_flag = false;
                  }
                  if (tmp == ' ') {
                        if (flag == true && ws_flag == false) count_ws += 1;
                        ws_flag = true;
                  }
                  if (tmp == '\n') {
                        if (flag == true && count_ws != 0) {
                              size++;
                              if (count_ws == 1) p << "Данные корректны" << '\n';
                              if (count_ws > 1) p << "Количество чисел в строке
больше 2" << '\n';
                        }
                        else {
```

```
                                    if (flag == false) p << "Обнаружена пустая строка"
<< '\n';
                                    else p << "Недостаточно данных" << '\n';
                              }
                              count_ws = 0; flag = false; ws_flag = false;
                        }
                        if (f.eof()) {
                              if (flag == true && count_ws == 1) {
                                    size++;
                                    p << "Данные корректны" << '\n';
                              }
                              break;
                        }
                        if (size == a) break;
                  } while (f.eof() != true);
                  if (size < a) a = size;
                  p << '\n';
                  f.close();
            }
            else {
                  if (f.is_open() == false) std::cout << "Не удалось открыть входной
файл";
                  if (p.is_open() == false) std::cout << "Не удалось открыть файл
протокола";
            }
}

void InpArr(float A[n][2], unsigned a, std::fstream& p) {
      std::fstream g; g.open("in.txt", std::ios::in);

      if (g.is_open() && p.is_open()) {
            int num; char tmp;
            bool flag = false, ws_flag = false, break_flag = false;
            unsigned count_ws = 0;
            unsigned position;
            g >> num;
            p << a << '\n' << std::flush;
            unsigned i = 0;
            while (g.eof() != true) {
                  if (i == a) break;
                  position = g.tellg();
                  break_flag = false;
                  while (1) {
                        g >> std::noskipws >> tmp;
                        if (tmp != ' ' && tmp != '\n') {
                              flag = true;
                              ws_flag = false;
                        }
                        if (tmp == ' ') {
                              if (flag == true && ws_flag == false) count_ws += 1;
                              ws_flag = true;
                        }
                        if (tmp == '\n') {
                              if (count_ws != 0) {
                                    g.seekg(position, std::ios::beg);
                                    for (unsigned q = 0; q < 2; q++) {
                                          g >> std::skipws >> A[i][q];
                                          p << std::left << std::setw(4) <<
A[i][q] << ' ' << std::flush;
                                    }
                                    p << '\n';
                                    tmp = ' ';
                                    while (tmp != '\n') {
                                          g >> std::noskipws >> tmp;
                                          if (g.eof()) break;
```

```
                                        }
                                        i++;
                                        break_flag = true;
                                }
                                else break_flag = true;
                        }
                        if (break_flag == true) {
                                count_ws = 0;
                                break;
                        }
                }
        }
        p << '\n';
        g.close();
    }
    else {
        if (g.is_open() == false) std::cout << "Не удалось открыть входной
файл";
        if (p.is_open() == false) std::cout << "Не удалось открыть файл
протокола";
    }
}



void PrintPoints(float A[n][2], unsigned i1, unsigned j1, unsigned i2, unsigned j2,
std::fstream& p) {
    std::string points1 = "(" + ((A[i1][0] == int(A[i1][0])) ?
std::to_string(int(A[i1][0])) : std::to_string(int(A[i1][0])) + '.' +
std::to_string(A[i1][0] - int(A[i1][0])).substr((std::to_string(A[i1][0])[0] == '-'
? 3 : 2), 2)) + ", " + ((A[i1][1] == int(A[i1][1])) ? std::to_string(int(A[i1][1]))
: std::to_string(int(A[i1][1])) + '.' + std::to_string(A[i1][1] -
int(A[i1][1])).substr((std::to_string(A[i1][1])[0] == '-' ? 3 : 2), 2)) + "), (" +
((A[j1][0] == int(A[j1][0])) ? std::to_string(int(A[j1][0])) :
std::to_string(int(A[j1][0])) + '.' + std::to_string(A[j1][0] -
int(A[j1][0])).substr((std::to_string(A[j1][0])[0] == '-' ? 3 : 2), 2)) + ", " +
((A[j1][1] == int(A[j1][1])) ? std::to_string(int(A[j1][1])) :
std::to_string(int(A[j1][1])) + '.' + std::to_string(A[j1][1] -
int(A[j1][1])).substr((std::to_string(A[j1][1])[0] == '-' ? 3 : 2), 2)) + ")";
    std::string points2 = "(" + ((A[i2][0] == int(A[i2][0])) ?
std::to_string(int(A[i2][0])) : std::to_string(int(A[i2][0])) + '.' +
std::to_string(A[i2][0] - int(A[i2][0])).substr((std::to_string(A[i2][0])[0] == '-'
? 3 : 2), 2)) + ", " + ((A[i2][1] == int(A[i2][1])) ? std::to_string(int(A[i2][1]))
: std::to_string(int(A[i2][1])) + '.' + std::to_string(A[i2][1] -
int(A[i2][1])).substr((std::to_string(A[i2][1])[0] == '-' ? 3 : 2), 2)) + "), (" +
((A[j2][0] == int(A[j2][0])) ? std::to_string(int(A[j2][0])) :
std::to_string(int(A[j2][0])) + '.' + std::to_string(A[j2][0] -
int(A[j2][0])).substr((std::to_string(A[j2][0])[0] == '-' ? 3 : 2), 2)) + ", " +
((A[j2][1] == int(A[j2][1])) ? std::to_string(int(A[j2][1])) :
std::to_string(int(A[j2][1])) + '.' + std::to_string(A[j2][1] -
int(A[j2][1])).substr((std::to_string(A[j2][1])[0] == '-' ? 3 : 2), 2)) + ")";
    p << points1 << std::setw(55 - points1.size()) << ' ' << points2 << '\n' <<
'\n';
}

bool PointsMemor(float A[n][2], unsigned i1, unsigned j1, unsigned i2, unsigned j2,
std::string B[10000], unsigned& k) {
    std::string tmp1 = "((" + std::to_string(A[i1][0]) + ", " +
std::to_string(A[i1][1]) + "), (" + std::to_string(A[j1][0]) + ", " +
std::to_string(A[j1][1]) + "); (" + std::to_string(A[i2][0]) + ", " +
std::to_string(A[i2][1]) + "), (" + std::to_string(A[j2][0]) + ", " +
std::to_string(A[j2][1]) + "))";
    std::string tmp2 = "((" + std::to_string(A[i2][0]) + ", " +
std::to_string(A[i2][1]) + "), (" + std::to_string(A[j2][0]) + ", " +
std::to_string(A[j2][1]) + "); (" + std::to_string(A[i1][0]) + ", " +
```

```
std::to_string(A[i1][1]) + "), (" + std::to_string(A[j1][0]) + ", " +
std::to_string(A[j1][1]) + "))";
	bool string_flag = false;
	for (unsigned i = 0; i < 10000; i++) {
		if (tmp1 == B[i] || tmp2 == B[i]) {
			string_flag = true;
			break;
		}
	}
	if (string_flag == true) return 0;
	else {
		B[k] = tmp1;
		k++;
		return 1;
	}
}

void Process(float A[n][2], unsigned a, std::fstream& p, std::fstream& e) {
	if (p.is_open() && e.is_open()) {
		float max_corn = 0;

		for (unsigned i1 = 0; i1 < a; i1++) {
			for (unsigned j1 = i1 + 1; j1 < a; j1++) {
				for (unsigned i2 = 0; i2 < a; i2++) {
					for (unsigned j2 = i2 + 1; j2 < a; j2++) {
						float k1, k2, b1, b2;
						Interpolation(A[i1][0], A[i1][1], A[j1][0],
A[j1][1], k1, b1);
						Interpolation(A[i2][0], A[i2][1], A[j2][0],
A[j2][1], k2, b2);
						float corn = abs((atan(k2) * (180 / pi) -
atan(k1) * (180 / pi))) <= 90 ? abs((atan(k2) * (180 / pi) - atan(k1) * (180 / pi)))
: 180 - abs((atan(k2) * (180 / pi) - atan(k1) * (180 / pi)));
						if (corn < 0) corn = 0;
						if (corn > max_corn) {
							max_corn = corn;
						}
					}
				}
			}
		}


std::string points_memor[10000];
		unsigned k = 0;
		for (unsigned i1 = 0; i1 < a; i1++) {
			for (unsigned j1 = i1 + 1; j1 < a; j1++) {
				for (unsigned i2 = 0; i2 < a; i2++) {
					for (unsigned j2 = i2 + 1; j2 < a; j2++) {
						if (PointsMemor(A, i1, j1, i2, j2,
points_memor, k)) {
							float k1, k2, b1, b2;
							Interpolation(A[i1][0], A[i1][1],
A[j1][0], A[j1][1], k1, b1);
							Interpolation(A[i2][0], A[i2][1],
A[j2][0], A[j2][1], k2, b2);
							float corn = abs((atan(k2) * (180 / pi)
- atan(k1) * (180 / pi))) <= 90 ? abs((atan(k2) * (180 / pi) - atan(k1) * (180 /
pi))) : 180 - abs((atan(k2) * (180 / pi) - atan(k1) * (180 / pi)));
							if (corn < 0) corn = 0;
							std::string s1, s2;
							if ((A[i1][0] - A[j1][0]) && (A[i2][0]
- A[j2][0])) {
								s1 = "Первая прямая: " +
std::to_string(int(k1)) + "." + std::to_string(k1 -
```

```
int(k1)).substr((std::to_string(k1)[0] == '-' ? 3 : 2), 2) + "x-1.00y" +
((std::to_string(b1)[0] == '-') ? "-" : "+") +
std::to_string(int(b1)).substr((std::to_string(b1)[0] == '-' &&
std::to_string(int(b1)) != "0") ? 1 : 0) + "." + std::to_string(b1 -
int(b1)).substr((std::to_string(b1)[0] == '-' ? 3 : 2), 2) + "=0";
                                                    s2 = "Вторая прямая: " +
std::to_string(int(k2)) + "." + std::to_string(k2 -
int(k2)).substr((std::to_string(k2)[0] == '-' ? 3 : 2), 2) + "x-1.00y" +
((std::to_string(b2)[0] == '-') ? "-" : "+") +
std::to_string(int(b2)).substr((std::to_string(b2)[0] == '-' &&
std::to_string(int(b2)) != "0") ? 1 : 0) + "." + std::to_string(b2 -
int(b2)).substr((std::to_string(b2)[0] == '-' ? 3 : 2), 2) + "=0";
                                                    p << s1 << std::setw(55 -
s1.size()) << " " << s2 << std::setw(55 - s2.size()) << std::right << "Угол: " <<
corn << "\n";
                                                        PrintPoints(A, i1, j1, i2, j2,
p);
                                                }
                                                else {
                                                    if (A[i1][0] - A[j1][0] == 0 &&
A[i2][0] - A[j2][0] == 0) {
                                                        s1 = "Первая прямая:
1.00x+0.00y" + std::string(((std::to_string(-1 * A[i1][0])[0] == '-') ? "-" : "+"))
+ std::to_string(int(-1 * A[i1][0])).substr((std::to_string(-1 * A[i1][0])[0] == '-'
&& -1 * A[i1][0] != 0 ? 1 : 0)) + "." + std::to_string(-1 * A[i1][0] - int(-1 *
A[i1][0])).substr((std::to_string(-1 * A[i1][0])[0] == '-' ? 3 : 2), 2) + "=0";
                                                        s2 = "Вторая прямая:
1.00x+0.00y" + std::string(((std::to_string(-1 * A[i2][0])[0] == '-') ? "-" : "+"))
+ std::to_string(int(-1 * A[i2][0])).substr((std::to_string(-1 * A[i2][0])[0] == '-'
&& -1 * A[i2][0] != 0 ? 1 : 0)) + "." + std::to_string(-1 * A[i2][0] - int(-1 *
A[i2][0])).substr((std::to_string(-1 * A[i2][0])[0] == '-' ? 3 : 2), 2) + "=0";
                                                        p << s1 << std::setw(55 -
s1.size()) << " " << s2 << std::setw(55 - s2.size()) << std::right << "Угол: " <<
corn << "\n";
                                                        PrintPoints(A, i1, j1, i2,
j2, p);
                                                    }
                                                    else if (A[i1][0] - A[j1][0] ==
0) {
                                                        s1 = "Первая прямая:
1.00x+0.00y" + std::string(((std::to_string(-1 * A[i1][0])[0] == '-') ? "-" : "+"))
+ std::to_string(int(-1 * A[i1][0])).substr((std::to_string(-1 * A[i1][0])[0] == '-'
&& -1 * A[i1][0] != 0 ? 1 : 0)) + "." + std::to_string(-1 * A[i1][0] - int(-1 *
A[i1][0])).substr((std::to_string(-1 * A[i1][0])[0] == '-' ? 3 : 2), 2) + "=0";
                                                        s2 = "Вторая прямая: " +
std::to_string(int(k2)) + "." + std::to_string(k2 -
int(k2)).substr((std::to_string(k2)[0] == '-' ? 3 : 2), 2) + "x-1.00y" +
((std::to_string(b2)[0] == '-') ? "-" : "+") +
std::to_string(int(b2)).substr((std::to_string(b2)[0] == '-' &&
std::to_string(int(b2)) != "0") ? 1 : 0) + "." + std::to_string(b2 -
int(b2)).substr((std::to_string(b2)[0] == '-' ? 3 : 2), 2) + "=0";
                                                        p << s1 << std::setw(55 -
s1.size()) << " " << s2 << std::setw(55 - s2.size()) << std::right << "Угол: " <<
corn << "\n";
                                                        PrintPoints(A, i1, j1, i2,
j2, p);
                                                    }
                                                    else if (A[i2][0] - A[j2][0] ==
0) {
                                                        s1 = "Первая прямая: " +
std::to_string(int(k1)) + "." + std::to_string(k1 -
int(k1)).substr((std::to_string(k1)[0] == '-' ? 3 : 2), 2) + "x-1.00y" +
((std::to_string(b1)[0] == '-') ? "-" : "+") +
std::to_string(int(b1)).substr((std::to_string(b1)[0] == '-' &&
```

```
std::to_string(int(b1)) != "0") ? 1 : 0) + "." + std::to_string(b1 -
int(b1)).substr((std::to_string(b1)[0] == '-' ? 3 : 2), 2) + "=0";
                                                        s2 = "Вторая прямая:
1.00x+0.00y" + std::string(((std::to_string(-1 * A[i2][0])[0] == '-') ? "-" : "+"))
+ std::to_string(int(-1 * A[i2][0])).substr((std::to_string(-1 * A[i2][0])[0] == '-'
&& -1 * A[i2][0] != 0 ? 1 : 0)) + "." + std::to_string(-1 * A[i2][0] - int(-1 *
A[i2][0])).substr((std::to_string(-1 * A[i2][0])[0] == '-' ? 3 : 2), 2) + "=0";
                                                        p << s1 << std::setw(55 -
s1.size()) << " " << s2 << std::setw(55 - s2.size()) << std::right << "Угол: " <<
corn << "\n";
                                                        PrintPoints(A, i1, j1, i2,
j2, p);
                                                }
                                        }
                                        if (corn == max_corn) {
                                                if ((A[i1][0] - A[j1][0]) &&
(A[i2][0] - A[j2][0])) {
                                                        s1 = "Первая прямая: " +
std::to_string(int(k1)) + "." + std::to_string(k1 -
int(k1)).substr((std::to_string(k1)[0] == '-' ? 3 : 2), 2) + "x-1.00y" +
((std::to_string(b1)[0] == '-') ? "-" : "+") +
std::to_string(int(b1)).substr((std::to_string(b1)[0] == '-' &&
std::to_string(int(b1)) != "0") ? 1 : 0) + "." + std::to_string(b1 -
int(b1)).substr((std::to_string(b1)[0] == '-' ? 3 : 2), 2) + "=0";
                                                        s2 = "Вторая прямая: " +
std::to_string(int(k2)) + "." + std::to_string(k2 -
int(k2)).substr((std::to_string(k2)[0] == '-' ? 3 : 2), 2) + "x-1.00y" +
((std::to_string(b2)[0] == '-') ? "-" : "+") +
std::to_string(int(b2)).substr((std::to_string(b2)[0] == '-' &&
std::to_string(int(b2)) != "0") ? 1 : 0) + "." + std::to_string(b2 -
int(b2)).substr((std::to_string(b2)[0] == '-' ? 3 : 2), 2) + "=0";
                                                        e << s1 << std::setw(55 -
s1.size()) << " " << s2 << std::setw(55 - s2.size()) << "Угол: " << corn << "\n";
                                                        PrintPoints(A, i1, j1, i2,
j2, e);
                                                }
                                                else {
                                                        if (A[i1][0] - A[j1][0] ==
0 && A[i2][0] - A[j2][0] == 0) {
                                                                s1 = "Первая прямая:
1.00x+0.00y" + std::string(((std::to_string(-1 * A[i1][0])[0] == '-') ? "-" : "+"))
+ std::to_string(int(-1 * A[i1][0])).substr((std::to_string(-1 * A[i1][0])[0] == '-'
&& -1 * A[i1][0] != 0 ? 1 : 0)) + "." + std::to_string(-1 * A[i1][0] - int(-1 *
A[i1][0])).substr((std::to_string(-1 * A[i1][0])[0] == '-' ? 3 : 2), 2) + "=0";
                                                                s2 = "Вторая прямая:
1.00x+0.00y" + std::string(((std::to_string(-1 * A[i2][0])[0] == '-') ? "-" : "+"))
+ std::to_string(int(-1 * A[i2][0])).substr((std::to_string(-1 * A[i2][0])[0] == '-'
&& -1 * A[i2][0] != 0 ? 1 : 0)) + "." + std::to_string(-1 * A[i2][0] - int(-1 *
A[i2][0])).substr((std::to_string(-1 * A[i2][0])[0] == '-' ? 3 : 2), 2) + "=0";
                                                                e << s1 <<
std::setw(55 - s1.size()) << " " << s2 << std::setw(55 - s2.size()) << std::right <<
"Угол: " << corn << "\n";
                                                                PrintPoints(A, i1,
j1, i2, j2, e);
                                                        }
                                                        else if (A[i1][0] -
A[j1][0] == 0) {
                                                                s1 = "Первая прямая:
1.00x+0.00y" + std::string(((std::to_string(-1 * A[i1][0])[0] == '-') ? "-" : "+"))
+ std::to_string(int(-1 * A[i1][0])).substr((std::to_string(-1 * A[i1][0])[0] == '-'
&& -1 * A[i1][0] != 0 ? 1 : 0)) + "." + std::to_string(-1 * A[i1][0] - int(-1 *
A[i1][0])).substr((std::to_string(-1 * A[i1][0])[0] == '-' ? 3 : 2), 2) + "=0";
                                                                s2 = "Вторая прямая:
" + std::to_string(int(k2)) + "." + std::to_string(k2 -
int(k2)).substr((std::to_string(k2)[0] == '-' ? 3 : 2), 2) + "x-1.00y" +
```

```
((std::to_string(b2)[0] == '-') ? "-" : "+") +
std::to_string(int(b2)).substr((std::to_string(b2)[0] == '-' &&
std::to_string(int(b2)) != "0") ? 1 : 0) + "." + std::to_string(b2 -
int(b2)).substr((std::to_string(b2)[0] == '-' ? 3 : 2), 2) + "=0";
                                                                e << s1 <<
std::setw(55 - s1.size()) << " " << s2 << std::setw(55 - s2.size()) << std::right <<
"Угол: " << corn << "\n";
                                                                PrintPoints(A, i1,
j1, i2, j2, e);
                                                        }
                                                        else if (A[i2][0] -
A[j2][0] == 0) {
                                                                s1 = "Первая прямая:
" + std::to_string(int(k1)) + "." + std::to_string(k1 -
int(k1)).substr((std::to_string(k1)[0] == '-' ? 3 : 2), 2) + "x-1.00y" +
((std::to_string(b1)[0] == '-') ? "-" : "+") +
std::to_string(int(b1)).substr((std::to_string(b1)[0] == '-' &&
std::to_string(int(b1)) != "0") ? 1 : 0) + "." + std::to_string(b1 -
int(b1)).substr((std::to_string(b1)[0] == '-' ? 3 : 2), 2) + "=0";
                                                                s2 = "Вторая прямая:
1.00x+0.00y" + std::string(((std::to_string(-1 * A[i2][0])[0] == '-') ? "-" : "+"))
+ std::to_string(int(-1 * A[i2][0])).substr((std::to_string(-1 * A[i2][0])[0] == '-'
&& -1 * A[i2][0] != 0 ? 1 : 0)) + "." + std::to_string(-1 * A[i2][0] - int(-1 *
A[i2][0])).substr((std::to_string(-1 * A[i2][0])[0] == '-' ? 3 : 2), 2) + "=0";
                                                                e << s1 <<
std::setw(55 - s1.size()) << " " << s2 << std::setw(55 - s2.size()) << std::right <<
"Угол: " << corn << "\n";
                                                                PrintPoints(A, i1,
j1, i2, j2, e);
                                                        }
                                                }
                                        }
                                }
                        }
                }
        }
        else {
                if (e.is_open() == false) std::cout << "Не удалось открыть выходной
файл";
                if (p.is_open() == false) std::cout << "Не удалось открыть файл
протокола";
        }
}

int main()
{
        setlocale(LC_ALL, "Russian");
        std::fstream p; p.open("Protocol.txt", std::ios::out);
        std::fstream e; e.open("out.txt", std::ios::out);
        float A[n][2];
        unsigned a;
        SizeArr(a, p);
        InpArr(A, a, p);
        Process(A, a, p, e);
        p.close();
        e.close();
}
```

11. ТЕСТОВЫЕ ПРИМЕРЫ

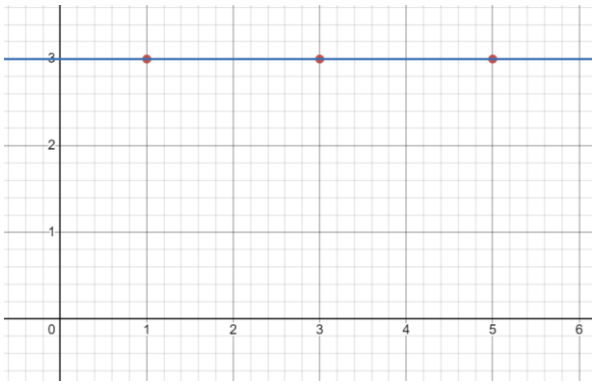

Рисунок 14. Все точки находятся на одной прямой

На рисунке 14 изображён случай, когда все точки находятся на одной прямой, то есть наибольший и единственный возможный угол будет равен 0°.

```
3
1 3
3 3
5 3
```

Рисунок 15. Входные данные в файле in.txt

```
Первая прямая: 0.00x-1.00y+3.00=0 Вторая прямая: 0.00x-1.00y+3.00=0 Угол: 0
(1, 3), (3, 3)                    (1, 3), (3, 3)

Первая прямая: 0.00x-1.00y+3.00=0 Вторая прямая: 0.00x-1.00y+3.00=0 Угол: 0
(1, 3), (3, 3)                    (1, 3), (5, 3)

Первая прямая: 0.00x-1.00y+3.00=0 Вторая прямая: 0.00x-1.00y+3.00=0 Угол: 0
(1, 3), (3, 3)                    (3, 3), (5, 3)

Первая прямая: 0.00x-1.00y+3.00=0 Вторая прямая: 0.00x-1.00y+3.00=0 Угол: 0
(1, 3), (5, 3)                    (1, 3), (5, 3)

Первая прямая: 0.00x-1.00y+3.00=0 Вторая прямая: 0.00x-1.00y+3.00=0 Угол: 0
(1, 3), (5, 3)                    (3, 3), (5, 3)

Первая прямая: 0.00x-1.00y+3.00=0 Вторая прямая: 0.00x-1.00y+3.00=0 Угол: 0
(3, 3), (5, 3)                    (3, 3), (5, 3)
```

Рисунок 16. Результат в файле out.txt

```
Данные корректны
Данные корректны
Данные корректны

3
1    3
3    3
5    3

Первая прямая: 0.00x-1.00y+3.00=0                Вторая прямая: 0.00x-1.00y+3.00=0                Угол: 0
(1, 3), (3, 3)                                   (1, 3), (3, 3)

Первая прямая: 0.00x-1.00y+3.00=0                Вторая прямая: 0.00x-1.00y+3.00=0                Угол: 0
(1, 3), (3, 3)                                   (1, 3), (5, 3)

Первая прямая: 0.00x-1.00y+3.00=0                Вторая прямая: 0.00x-1.00y+3.00=0                Угол: 0
(1, 3), (3, 3)                                   (3, 3), (5, 3)

Первая прямая: 0.00x-1.00y+3.00=0                Вторая прямая: 0.00x-1.00y+3.00=0                Угол: 0
(1, 3), (5, 3)                                   (1, 3), (5, 3)

Первая прямая: 0.00x-1.00y+3.00=0                Вторая прямая: 0.00x-1.00y+3.00=0                Угол: 0
(1, 3), (5, 3)                                   (3, 3), (5, 3)

Первая прямая: 0.00x-1.00y+3.00=0                Вторая прямая: 0.00x-1.00y+3.00=0                Угол: 0
(3, 3), (5, 3)                                   (3, 3), (5, 3)
```

Рисунок 17. Файл протокола

А теперь рассмотрим ситуацию, где точки являются вершинами параллелограмма и нет перпендикулярных прямых. Попробуем вручную найти угол между одной из пар прямых и сравним результат с программой.

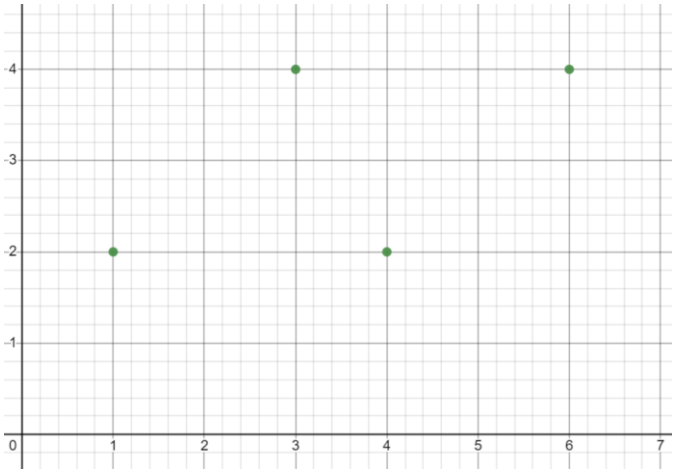

Рисунок 18. Точки расположены как вершины параллелограмма

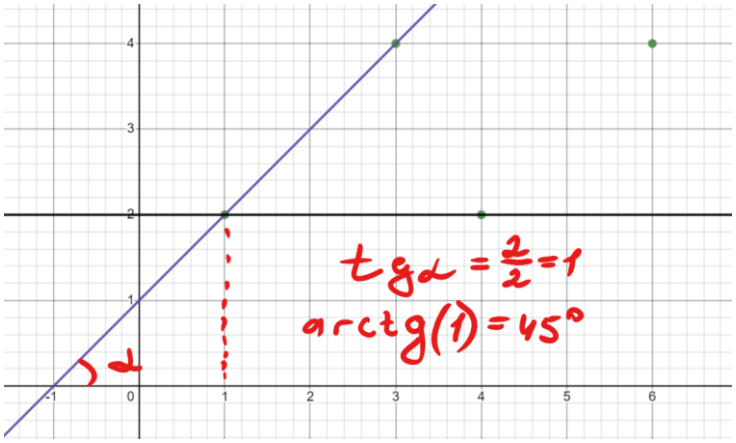

Рисунок 19. Прямые 1x-1y+1=0 и 0x-1y-2=0

На рисунке 19 видно, что одна из прямых параллельна оси Ox, то есть углом между этими прямыми будет просто арктангенс от углового коэффициента другой прямой. Угол между прямыми оказался равен 45°.

```
4
1 2
4 2
6 4
3 4
```

Рисунок 20. Входные данные в файле in.txt

```
Первая прямая: 0.40x-1.00y+1.60=0
(1, 2), (6, 4)
Вторая прямая: -2.00x-1.00y+10.00=0
(4, 2), (3, 4)
Угол: 85.2364
```

Рисунок 21. Результат в файле out.txt

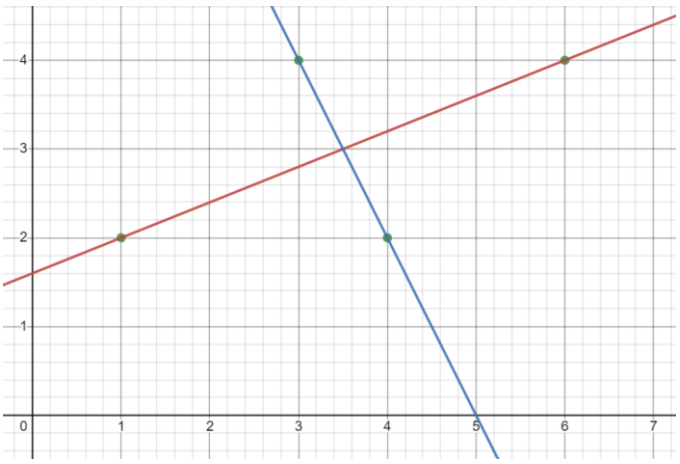

Рисунок 22. Полученная пара прямых в desmos

Результатом оказалась пара прямых, являющихся диагоналями параллелограмма, но в файле протокола можно найти пару прямых, для которой мы вручную посчитали угол, и сравнить значения.

```
Данные корректны
Данные корректны
Данные корректны
Данные корректны

4
1    2
4    2
6    4
3    4

Первая прямая: 0.00x-1.00y+2.00=0                Вторая прямая: 0.00x-1.00y+2.00=0                Угол: 0
(1, 2), (4, 2)                                   (1, 2), (4, 2)

Первая прямая: 0.00x-1.00y+2.00=0                Вторая прямая: 0.40x-1.00y+1.60=0                Угол: 21.8014
(1, 2), (4, 2)                                   (1, 2), (6, 4)

Первая прямая: 0.00x-1.00y+2.00=0                Вторая прямая: 1.00x-1.00y+1.00=0                Угол: 45
(1, 2), (4, 2)                                   (1, 2), (3, 4)

Первая прямая: 0.00x-1.00y+2.00=0                Вторая прямая: 1.00x-1.00y-2.00=0                Угол: 45
(1, 2), (4, 2)                                   (4, 2), (6, 4)

Первая прямая: 0.00x-1.00y+2.00=0                Вторая прямая: -2.00x-1.00y+10.00=0              Угол: 63.435
(1, 2), (4, 2)                                   (4, 2), (3, 4)

Первая прямая: 0.00x-1.00y+2.00=0                Вторая прямая: 0.00x-1.00y+4.00=0                Угол: 0
(1, 2), (4, 2)                                   (6, 4), (3, 4)

Первая прямая: 0.40x-1.00y+1.60=0                Вторая прямая: 0.40x-1.00y+1.60=0                Угол: 0
(1, 2), (6, 4)                                   (1, 2), (6, 4)
```

Рисунок 23. Часть файла протокола с отмеченной парой прямых

Значения совпали, угол действительно оказался равен 45°. Теперь возьмём произвольные 10 точек и на их примере покажем метод нахождения угла.

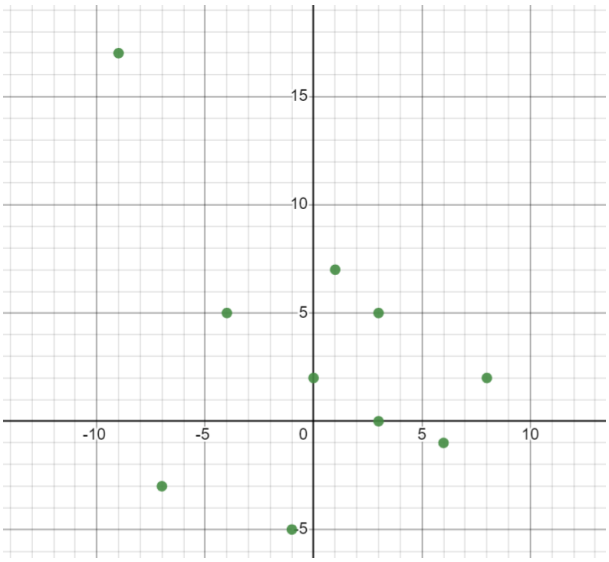

Рисунок 24. Произвольный набор точек

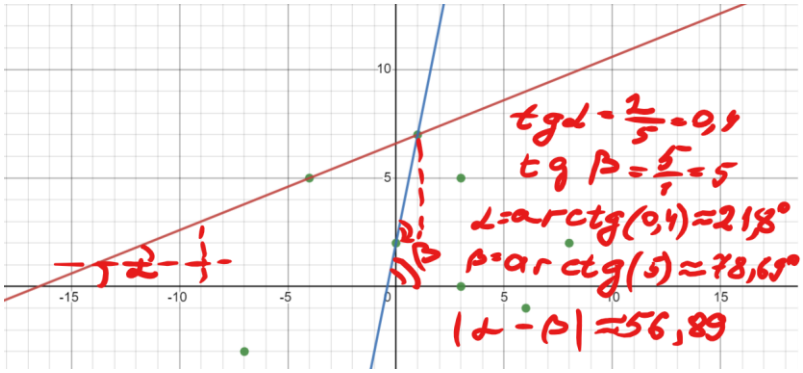

Рисунок 25. Прямые 0.4x-1y+6.6=0 и 5x-1y+2=0

Как видно на рисунке 25 угол получился равным приблизительно 56.89°. Запустим программу и найдём эту пару прямых в файле протокола.

```
10

  1    7

8
-4 5

6 -1 9
3 0

0 2
8 2

-7 -3
-1 -5

3 5
```

Рисунок 26. Входные данные с лишними пробелами, пустыми строчками, одними значением в строке или большем 2-ух в файле in.txt

```
Обнаружена пустая строка
Данные корректны
Обнаружена пустая строка
Недостаточно данных
Данные корректны
Обнаружена пустая строка
Количество чисел в строке больше 2
Данные корректны
Обнаружена пустая строка
Данные корректны
Данные корректны
Обнаружена пустая строка
Данные корректны
Данные корректны
Обнаружена пустая строка
Данные корректны
Данные корректны

10
1    7
-4   5
6    -1
3    0
0    2
8    2
-7   -3
-1   -5
3    5
39   17

Первая прямая: 0.40x-1.00y+6.60=0                Вторая прямая: 0.40x-1.00y+6.60=0                Угол: 0
(1, 7), (-4, 5)                                  (1, 7), (-4, 5)

Первая прямая: 0.40x-1.00y+6.60=0                Вторая прямая: -1.60x-1.00y+8.60=0               Угол: 79.796
(1, 7), (-4, 5)                                  (1, 7), (6, -1)

Первая прямая: 0.40x-1.00y+6.60=0                Вторая прямая: -3.50x-1.00y+10.50=0              Угол: 84.144
(1, 7), (-4, 5)                                  (1, 7), (3, 0)

Первая прямая: 0.40x-1.00y+6.60=0                Вторая прямая: 5.00x-1.00y+2.00=0                Угол: 56.8887
(1, 7), (-4, 5)                                  (1, 7), (0, 2)
```

Рисунок 27. Файл протокола с отмеченной парой прямых

Как видно на рисунке 27 угол оказался равен 56.8887°, что даже более точно, чем значение, полученное вручную. Также на рисунке 26 можно заметить, что во входном файле были добавлены лишние пробелы, пустые строки и ситуации с большим или меньшим 2 количеством чисел в строке. Все эти случаи отражены в файле протокола и взяты лишь нужные значения.

12. ВЫВОДЫ

В ходе выполнения курсовой работы я научился работать с положением каретки в файле, а также со строковым типом данных, что позволило оформить приятный для чтения вывод в результирующий файл и файл протокола. В результате получилась программа, которая выполняет поставленную задачу с использованием двухмерного массива и успешно справляется с различными возможными ситуациями.