Минобрнауки РФ

Санкт-Петербургский государственный электротехнический университет

им. В.И. Ульянова (Ленина) «ЛЭТИ»

(СПбГЭТУ)

Факультет компьютерных технологий и информатики

Кафедра САПР

КУРСОВАЯ РАБОТА

По дисциплине «Программирование»

На тему «Решение геометрических задач на плоскости»

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Калмычков В. А.

Студент гр. 1309: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фомин В. В.

Санкт-Петербург

2020

# Аннотация

В данной курсовой работе рассмотрено решение геометрической задачи с использование массивов и чтением из файла. Приводятся описания как математического решения данной задачи, так и алгоритма решения на компьютере. Задание заключается в построении всех возможных треугольниках и нахождении площади их пересечения.

# Summary

In this course work, the solution of a geometric problem with the use of arrays and reading from a file is considered. Descriptions of both the mathematical solution of this problem and the algorithm for solving it on a computer are given. The task is to build all possible triangles and find the area of their intersection.

Оглавление

[Формулировка задания 4](#_Toc533079549)

[Цель работы 4](#_Toc533079550)

[Математическая постановка задачи 4](#_Toc533079551)

[Теоретические положения 4](#_Toc533079555)

[Организация диалога с пользователем 5](#_Toc533079556)

[Анализ задания и контрольный пример 5](#_Toc533079557)

[Постановка задачи для решения на компьютере 8](#_Toc533079558)

[Определение внутреннего формата представления данных 8](#_Toc533079559)

[Описание алгоритма 9](#_Toc533079562)

[Блок – схемы 10](#_Toc533079563)

[Текст программы 15](#_Toc533079568)

[Примеры работы программы 20](#_Toc533079569)

[Вывод 22](#_Toc533079570)

[Список литературы 23](#_Toc533079571)

# Формулировка задания

Дано N точек на плоскости. Найти пары треугольников, построенных из этих точек, имеющие наибольшую площадь пересечения.

# Цель работы

Обобщить знания и практические навыки по программированию, полученные за первый семестр.

# Математическая постановка задачи

Дано: Дан файл с координатами точек.

Найти: Площадь пересечения для каждой пары треугольников. Определить максимальную площадь пересечения и выбрать пары треугольников с такой площадью.

Способ решения: Считать массив точек. Из точек построить все возможные треугольники. Из треугольников построить все возможные пары. Для каждой пары треугольников найти площадь их пересечения. Находим максимальную площадь среди всех пар. Находим все пары с такой площадью.

# Теоретические положения

Нужно найти площадь пересечения 2 треугольников. Для этого определим точки для каждого треугольника в паре, лежащие внутри другого, а также точки пересечения сторон. Всего может быть до 6 таких точек. Находим току со среднеарифметическими координатами от всех точек. Далее сортируем точки по их полярному углу относительно этой точки. Далее находим площадь полученного многоугольника (например, через определитель), по порядку беря для этого 2 последовательные точки из отсортированного массива и среднюю точку.

Для определения принадлежности точки треугольнику нужно найти косое произведение (фактически - определитель) между всеми парами сторон. Если оно во всех 3 случаях >= 0 или во всех 3 случаях <= 0 , то точка принадлежит, иначе – нет.

# **Организация диалога с пользователем**

Для корректной работы программы, пользователю надо ввести в файлы ввода числа, являющиеся координатами точек. После анализа считанных данных, в протокол запишется информация о:

1. Считанных точках.
2. Всех построенных треугольниках.
3. Всех парах треугольников и их площадью пересечения.

В выходной файл будет записан результат: максимальная площадь пересечения в паре и список таких пар.

# Анализ задания и контрольный пример

Для успешного выполнения программы нам понадобится как минимум 4 точки. Для контрольного примера следующие:

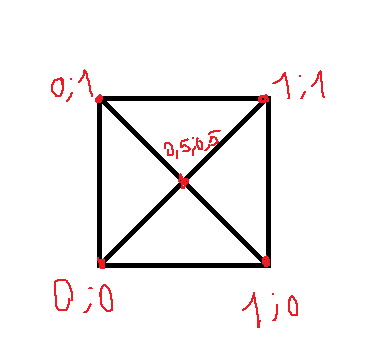
0 0

0 1

1 0

1 1

При построении треугольников на плоскости, мы наблюдаем следующую картину:



**Протокол:**

выбранные точки  
0 0  
0 1  
1 0  
1 1  
  
выбранные треугольники  
t0: ( 0; 0 ) ( 0; 1 ) ( 1; 0 ) s=0.5  
t1: ( 0; 0 ) ( 0; 1 ) ( 1; 1 ) s=0.5  
t2: ( 0; 0 ) ( 1; 0 ) ( 1; 1 ) s=0.5  
t3: ( 0; 1 ) ( 1; 0 ) ( 1; 1 ) s=0.5  
  
пары треугольников и площадь  
t0, t1. S=0.25  
t0, t2. S=0.25  
t0, t3. S=0  
t1, t2. S=0  
t1, t3. S=0.25  
t2, t3. S=0.25

**Выходной файл:**

максимальная площадь: 0.25  
пары треугольников с такой площадью:  
t1: ( 0; 0 ) ( 0; 1 ) ( 1; 0 ) t2: ( 0; 0 ) ( 0; 1 ) ( 1; 1 )   
t1: ( 0; 0 ) ( 0; 1 ) ( 1; 0 ) t2: ( 0; 0 ) ( 1; 0 ) ( 1; 1 )   
t1: ( 0; 0 ) ( 0; 1 ) ( 1; 1 ) t2: ( 0; 1 ) ( 1; 0 ) ( 1; 1 )   
t1: ( 0; 0 ) ( 1; 0 ) ( 1; 1 ) t2: ( 0; 1 ) ( 1; 0 ) ( 1; 1 )

# Постановка задачи для решения на компьютере

Используемые библиотеки: <iostream>, <string>, <fstream>. Координаты точек это вещественные числа типа double. Считывание координат производится из файлов формата .txt. Точкой считается любой считаный набор чисел с соответственными индексами, где существуют *x* и *y*. Протокол с информацией о ходе работы программы записывается в файл формата .txt, а конечный результат выводится в выходной файл. Программа предусматривает все случаи обработки данных из входных файлов.

# Определение внутреннего формата представления данных

Программа считывает координаты из файла и записывает их в двумерный массив a. Все треугольники лежат в массиве trs. Все пары лежат в массиве pairs. Для треугольников имеется структура triangle, а для пар треугольников pair.

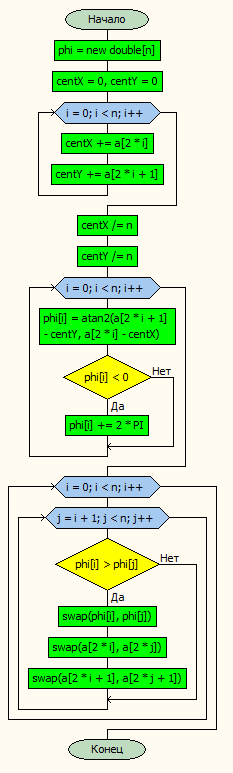
## Функции программы:

|  |  |
| --- | --- |
| **Имя функции** | **Назначение** |
|
| dotPr | Скалярное произведение. |
| det | Определитель. |
| sortVertexes | Сортировка точек выпуклой фигуры по полярному углу. |
| isw | Проверка символа на пробел или null. |
| isd | Проверка символа на число или минус. |
| sizeOfElements | Количество элементов в строке. |
| toDoubleArray | Достать из строки несколько вещественных чисел и вернуть их в массиве |
| inTriangle | Принадлежность точки треугольнику |
| intersect | Проверка пересечения отрезков и, если отрезки пересекаются нахождение точки пересечения. |
| square | Нахождение площади пересечения двух треугольников. |
| main | Вход в программу. |

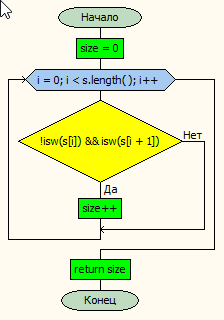
## 

# Блок – схемы

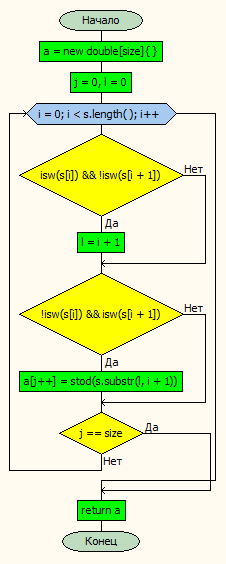
## Sort vertexes:



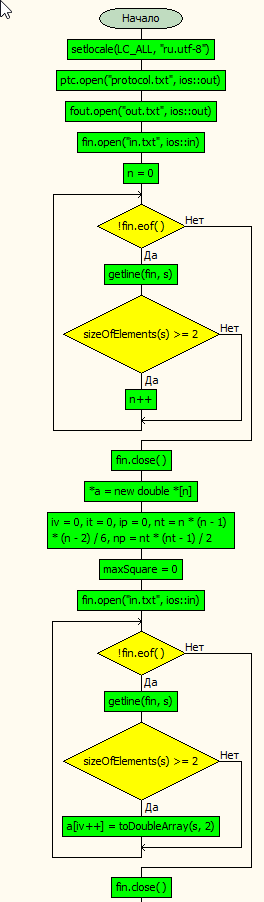
## sizeOfElements:

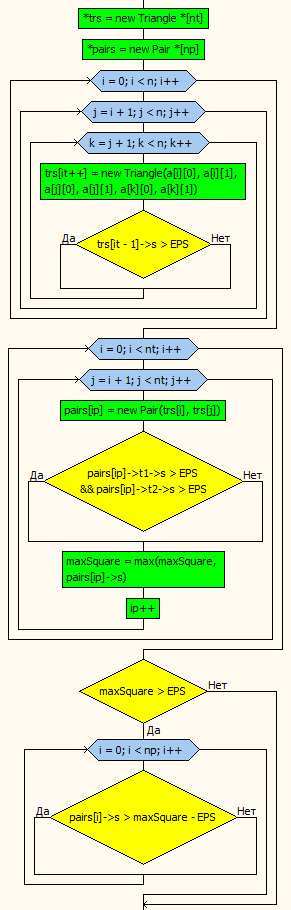


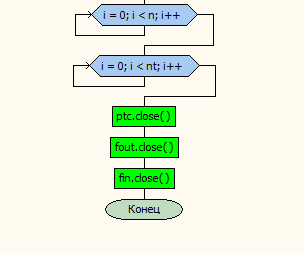
## toDoubleArray:



## Int main:







# Текст программы

#include <fstream>  
  
using namespace std;  
  
const double PI = 3.14159265358979323846264338327950288419716939937510, EPS = 0.000001;  
  
double dotPr(double x1, double y1, double x2, double y2) {  
 return x1 \* x2 + y1 \* y2;  
}  
  
double det(double x1, double y1, double x2, double y2) {  
 return x1 \* y2 - y1 \* x2;  
}  
  
struct Triangle {  
 double x1, y1, x2, y2, x3, y3, s;  
  
 Triangle(double x1, double y1, double x2, double y2, double x3, double y3) {  
 this->x1 = x1;  
 this->x2 = x2;  
 this->x3 = x3;  
 this->y1 = y1;  
 this->y2 = y2;  
 this->y3 = y3;  
 s = abs(det(x1-x3, y1-y3, x2-x3, y2-y3) / 2);  
 }  
};  
  
void sortVertexes(double \*a, int n) {  
 double \*phi = new double[n];  
 double centX = 0, centY = 0;  
 for (int i = 0; i < n; i++) {  
 centX += a[2 \* i];  
 centY += a[2 \* i + 1];  
 }  
 centX /= n;  
 centY /= n;  
 for (int i = 0; i < n; i++) {  
 phi[i] = atan2(a[2 \* i + 1] - centY, a[2 \* i] - centX);  
 if (phi[i] < 0) {  
 phi[i] += 2 \* PI;  
 }  
 }  
 for (int i = 0; i < n; i++) {  
 for (int j = i + 1; j < n; j++) {  
 if (phi[i] > phi[j]) {  
 swap(phi[i], phi[j]);  
 swap(a[2 \* i], a[2 \* j]);  
 swap(a[2 \* i + 1], a[2 \* j + 1]);  
 }  
 }  
 }  
 delete[] phi;  
}  
  
bool isd(int x) {  
 return x > 47 && x < 58 || x == 45;  
}  
  
bool isw(int x) {  
 return x == 0 || x == 32;  
}  
  
  
int sizeOfElements(string s) {  
 int size = 0;  
 for (int i = 0; i < s.length(); i++) {  
 if (!isw(s[i]) && isw(s[i + 1])) {  
 size++;  
 }  
 }  
 return size;  
}  
  
  
double \*toDoubleArray(string s, int size) {  
 double \*a = new double[size]{};  
 int j = 0, l = 0;  
 for (int i = 0; i < s.length(); i++) {  
 if (isw(s[i]) && !isw(s[i + 1])) {  
 l = i + 1;  
 }  
 if (!isw(s[i]) && isw(s[i + 1])) {  
 a[j++] = stod(s.substr(l, i + 1));  
 }  
 if (j == size) {  
 break;  
 }  
 }  
 return a;  
}  
  
  
bool inTriangle(Triangle t, double pointX, double pointY) {  
 double k = det(t.x1 - pointX, t.y1 - pointY, t.x2 - t.x1, t.y2 - t.y1);  
 double m = det(t.x2 - pointX, t.y2 - pointY, t.x3 - t.x2, t.y3 - t.y2);  
 double n = det(t.x3 - pointX, t.y3 - pointY, t.x1 - t.x3, t.y1 - t.y3);  
 return (k >= 0 && m >= 0 && n >= 0) || (k <= 0 && m <= 0 && n <= 0);  
}  
  
bool intersect(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4, double &xr,  
 double &yr) {  
 if (abs(det(x2 - x1, y2 - y1, x4 - x3, y4 - y3)) < EPS) {  
 return false;  
 }  
 xr = ((x2 - x1) \* det(x3, y3, x4, y4) - (x4 - x3) \* det(x1, y1, x2, y2)) / det(x2 - x1, y2 - y1, x4 - x3, y4 - y3);  
 yr = ((y2 - y1) \* det(x3, y3, x4, y4) - (y4 - y3) \* det(x1, y1, x2, y2)) / det(x2 - x1, y2 - y1, x4 - x3, y4 - y3);  
 if (dotPr(xr - x1, yr - y1, xr - x2, yr - y2) > 0 || dotPr(xr - x3, yr - y3, xr - x4, yr - y4) > 0) {  
 return false;  
 }  
 return true;  
}  
  
double square(Triangle t1, Triangle t2) {  
 double v[30];//в реале 6, но случаев 15. \* 2 за счет x, y.  
 int n = 0;  
 if (inTriangle(t1, t2.x1, t2.y1)) {  
 v[n++] = t2.x1;  
 v[n++] = t2.y1;  
 }  
 if (inTriangle(t1, t2.x2, t2.y2)) {  
 v[n++] = t2.x2;  
 v[n++] = t2.y2;  
 }  
 if (inTriangle(t1, t2.x3, t2.y3)) {  
 v[n++] = t2.x3;  
 v[n++] = t2.y3;  
 }  
 if (inTriangle(t2, t1.x1, t1.y1)) {  
 v[n++] = t1.x1;  
 v[n++] = t1.y1;  
 }  
 if (inTriangle(t2, t1.x2, t1.y2)) {  
 v[n++] = t1.x2;  
 v[n++] = t1.y2;  
 }  
 if (inTriangle(t2, t1.x3, t1.y3)) {  
 v[n++] = t1.x3;  
 v[n++] = t1.y3;  
 }  
 double ix, iy, cx = 0, cy = 0, s = 0;  
 if (intersect(t1.x1, t1.y1, t1.x2, t1.y2, t2.x1, t2.y1, t2.x2, t2.y2, ix, iy)) {  
 v[n++] = ix;  
 v[n++] = iy;  
 }  
 if (intersect(t1.x1, t1.y1, t1.x2, t1.y2, t2.x2, t2.y2, t2.x3, t2.y3, ix, iy)) {  
 v[n++] = ix;  
 v[n++] = iy;  
 }  
 if (intersect(t1.x1, t1.y1, t1.x2, t1.y2, t2.x1, t2.y1, t2.x3, t2.y3, ix, iy)) {  
 v[n++] = ix;  
 v[n++] = iy;  
 }  
 if (intersect(t1.x2, t1.y2, t1.x3, t1.y3, t2.x1, t2.y1, t2.x2, t2.y2, ix, iy)) {  
 v[n++] = ix;  
 v[n++] = iy;  
 }  
 if (intersect(t1.x2, t1.y2, t1.x3, t1.y3, t2.x2, t2.y2, t2.x3, t2.y3, ix, iy)) {  
 v[n++] = ix;  
 v[n++] = iy;  
 }  
 if (intersect(t1.x2, t1.y2, t1.x3, t1.y3, t2.x1, t2.y1, t2.x3, t2.y3, ix, iy)) {  
 v[n++] = ix;  
 v[n++] = iy;  
 }  
 if (intersect(t1.x1, t1.y1, t1.x3, t1.y3, t2.x1, t2.y1, t2.x2, t2.y2, ix, iy)) {  
 v[n++] = ix;  
 v[n++] = iy;  
 }  
 if (intersect(t1.x1, t1.y1, t1.x3, t1.y3, t2.x2, t2.y2, t2.x3, t2.y3, ix, iy)) {  
 v[n++] = ix;  
 v[n++] = iy;  
 }  
 if (intersect(t1.x1, t1.y1, t1.x3, t1.y3, t2.x1, t2.y1, t2.x3, t2.y3, ix, iy)) {  
 v[n++] = ix;  
 v[n++] = iy;  
 }  
  
 sortVertexes(v, n / 2);  
 for (int i = 0; i < n / 2; i++) {  
 cx += v[2 \* i];  
 cy += v[2 \* i + 1];  
 }  
 cx /= n / 2;  
 cy /= n / 2;  
 for (int i = 0; i < n / 2; i++) {  
 s += det(v[i \* 2] - cx, v[i \* 2 + 1] - cy, v[(i \* 2 + 2) % n] - cx, v[(i \* 2 + 3) % n] - cy) / 2;  
 }  
 return s;  
}  
  
struct Pair {  
 Triangle \*t1, \*t2;  
 double s;  
  
 Pair(Triangle \*t1, Triangle \*t2) {  
 this->t1 = t1;  
 this->t2 = t2;  
 s = square(\*t1, \*t2);  
 if (s < EPS && s > -EPS) {  
 s = 0;  
 }  
 }  
  
 ~Pair() {  
 delete t1;  
 delete t2;  
 }  
  
};  
  
int main() {  
 setlocale(LC\_ALL, "ru.utf-8");  
 fstream ptc;  
 fstream fout;  
 fstream fin;  
 ptc.open("protocol.txt", ios::out);  
 fout.open("out.txt", ios::out);  
 fin.open("in.txt", ios::in);  
 int n = 0;  
 string s;  
 while (!fin.eof()) {  
 getline(fin, s);  
 if (sizeOfElements(s) >= 2) {  
 n++;  
 }  
 }  
 fin.close();  
 double \*\*a = new double \*[n];  
 int iv = 0, it = 0, ip = 0, nt = n \* (n - 1) \* (n - 2) / 6, np = nt \* (nt - 1) / 2;  
 double maxSquare = 0;  
 fin.open("in.txt", ios::in);  
 ptc << "выбранные точки\n";  
 while (!fin.eof()) {  
 getline(fin, s);  
 if (sizeOfElements(s) >= 2) {  
 //ptc << s << "\n";  
 a[iv++] = toDoubleArray(s, 2);  
 ptc << a[iv - 1][0] << " " << a[iv - 1][1] << "\n";  
 }  
 }  
 fin.close();  
 Triangle \*\*trs = new Triangle \*[nt];  
 Pair \*\*pairs = new Pair \*[np];  
 ptc << "\nвыбранные треугольники\n";  
 for (int i = 0; i < n; i++) {  
 for (int j = i + 1; j < n; j++) {  
 for (int k = j + 1; k < n; k++) {  
 trs[it++] = new Triangle(a[i][0], a[i][1], a[j][0], a[j][1], a[k][0], a[k][1]);  
 if (trs[it - 1]->s > EPS) {  
 ptc << "t" << it - 1 << ": ( " << trs[it - 1]->x1 << "; " << trs[it - 1]->y1 << " ) " << "( "  
 << trs[it - 1]->x2 << "; " << trs[it - 1]->y2 << " ) " << "( " << trs[it - 1]->x3 << "; "  
 << trs[it - 1]->y3 << " ) s=" << trs[it - 1]->s << "\n";  
 }  
 }  
 }  
 }  
 ptc << "\nпары треугольников и площадь\n";  
 for (int i = 0; i < nt; i++) {  
 for (int j = i + 1; j < nt; j++) {  
 pairs[ip] = new Pair(trs[i], trs[j]);  
 if (pairs[ip]->t1->s > EPS && pairs[ip]->t2->s > EPS) {  
 ptc << "t" << i << ", t" << j << ". S=" << pairs[ip]->s << "\n";  
 }  
 maxSquare = max(maxSquare, pairs[ip]->s);  
 ip++;  
 }  
 }  
 //cout << "ip: "<< ip << " it: " << it <<"\n";  
 if (maxSquare > EPS) {  
 fout << "максимальная площадь: " << maxSquare << "\nпары треугольников с такой площадью:\n";  
 for (int i = 0; i < np; i++) {  
 if (pairs[i]->s > maxSquare - EPS) {  
 fout << "t1: ";  
 fout << "( " << pairs[i]->t1->x1 << "; " << pairs[i]->t1->y1 << " ) " << "( " << pairs[i]->t1->x2  
 << "; "  
 << pairs[i]->t1->y2 << " ) " << "( " << pairs[i]->t1->x3 << "; " << pairs[i]->t1->y3 << " ) ";  
 fout << " t2: ";  
 fout << "( " << pairs[i]->t2->x1 << "; " << pairs[i]->t2->y1 << " ) " << "( " << pairs[i]->t2->x2  
 << "; "  
 << pairs[i]->t2->y2 << " ) " << "( " << pairs[i]->t2->x3 << "; " << pairs[i]->t2->y3 << " ) ";  
 fout << "\n";  
 }  
 }  
 } else {  
 fout << "таких пересекающихся пар треугольников не существует\n";  
 }  
 for (int i = 0; i < n; i++) {  
 delete[] a[i];  
 }  
 delete[] a;  
 for (int i = 0; i < nt; i++) {  
 delete trs[i];  
 }  
 delete[] pairs;  
 ptc.close();  
 fout.close();  
 fin.close();  
}

# Примеры работы программы

**Файл in.txt:**

4 6  
3 3,35  
9 4,22 4  
5  
  
5 8,9  
2 0,654  
8,64 5,423

**Файл out.txt:**

максимальная площадь: 18.9622  
пары треугольников с такой площадью:  
t1: ( 9; 4.22 ) ( 5; 8.9 ) ( 2; 0.654 ) t2: ( 5; 8.9 ) ( 2; 0.654 ) ( 8.64; 5.423 )

**Файл protocol.txt:**

выбранные точки  
4 6  
3 3.35  
9 4.22  
5 8.9  
2 0.654  
8.64 5.423  
  
выбранные треугольники  
t0: ( 4; 6 ) ( 3; 3.35 ) ( 9; 4.22 ) s=7.515  
t1: ( 4; 6 ) ( 3; 3.35 ) ( 5; 8.9 ) s=0.125  
t2: ( 4; 6 ) ( 3; 3.35 ) ( 2; 0.654 ) s=0.023  
t3: ( 4; 6 ) ( 3; 3.35 ) ( 8.64; 5.423 ) s=6.4365  
t4: ( 4; 6 ) ( 9; 4.22 ) ( 5; 8.9 ) s=8.14  
t5: ( 4; 6 ) ( 9; 4.22 ) ( 2; 0.654 ) s=15.145  
t6: ( 4; 6 ) ( 9; 4.22 ) ( 8.64; 5.423 ) s=2.6871  
t7: ( 4; 6 ) ( 5; 8.9 ) ( 2; 0.654 ) s=0.227  
t8: ( 4; 6 ) ( 5; 8.9 ) ( 8.64; 5.423 ) s=7.0165  
t9: ( 4; 6 ) ( 2; 0.654 ) ( 8.64; 5.423 ) s=12.9797  
t10: ( 3; 3.35 ) ( 9; 4.22 ) ( 5; 8.9 ) s=15.78  
t11: ( 3; 3.35 ) ( 9; 4.22 ) ( 2; 0.654 ) s=7.653  
t12: ( 3; 3.35 ) ( 9; 4.22 ) ( 8.64; 5.423 ) s=3.7656  
t13: ( 3; 3.35 ) ( 5; 8.9 ) ( 2; 0.654 ) s=0.079  
t14: ( 3; 3.35 ) ( 5; 8.9 ) ( 8.64; 5.423 ) s=13.578  
t15: ( 3; 3.35 ) ( 2; 0.654 ) ( 8.64; 5.423 ) s=6.56622  
t16: ( 9; 4.22 ) ( 5; 8.9 ) ( 2; 0.654 ) s=23.512  
t17: ( 9; 4.22 ) ( 5; 8.9 ) ( 8.64; 5.423 ) s=1.5636  
t18: ( 9; 4.22 ) ( 2; 0.654 ) ( 8.64; 5.423 ) s=4.85238  
t19: ( 5; 8.9 ) ( 2; 0.654 ) ( 8.64; 5.423 ) s=20.2232  
  
пары треугольников и площадь  
t0, t1. S=0  
t0, t2. S=0.0113954  
t0, t3. S=4.74121  
t0, t4. S=0  
t0, t5. S=7.5036  
t0, t6. S=0  
t0, t7. S=0.0113954  
t0, t8. S=0  
t0, t9. S=6.6346  
t0, t10. S=7.515

………………………………………………………………………….

t15, t16. S=6.52931  
t15, t17. S=0.0369127  
t15, t18. S=0  
t15, t19. S=6.56622  
t16, t17. S=0  
t16, t18. S=4.54981  
t16, t19. S=18.9622  
t17, t18. S=0.302572  
t17, t19. S=1.26103  
t18, t19. S=0

# Вывод

После выполнения данной курсовой работы я структурировал знания по дисциплине «Программирование», научился применять все полученные мной знания за 1 семестр. Научился применять изученные библиотеки для решения различных задач.