УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №2.1

по предмету «Основы алгоритмизации и программирования»

Вариант 5

Выполнил:

Бражалович А. И.

Гр. 351005

Проверила:

Данилова Г. В.

Минск 2023

**Задание:**

Многоугольник задан координатами своих вершин. Найти площадь этого многоугольника.

**Код программы Delphi:**

Program Lab2\_1;

Uses

System.SysUtils;

Const

MIN = -10000;

MAX = 10000;

MIN\_N = 3;

MAX\_N = 10000;

Length\_Of\_Arr = 2;

Var

N, I, J: Integer;

Coordinates: Array Of Array Of Real;

Vector: Array[1..2] Of Array[1..2] Of Real;

IsCorrect, NotConvexPolygon, NotOnSameLine, IsCorrectFigure, SameCoordinate: Boolean;

VectorMultiplication, PrevVectorMultiplication, X\_Test, Max\_X, Min\_X, Max\_Y, Min\_Y, Mid\_X, Mid\_Y, SquareOfFigure: Real;

Begin

SquareOfFigure := 0;

N := 0;

PrevVectorMultiplication := 0;

NotConvexPolygon := True;

NotOnSameLine := True;

IsCorrectFigure := True;

Writeln('Эта программа находит площадь N-угольника.');

Repeat

IsCorrect := True;

Write('Введите количество вершин многоугольника [' , MIN\_N, '; ', MAX\_N, ']: ');

Try

Readln(N);

Except

Writeln('Проверьте корректность ввода данных!');

IsCorrect := False;

End;

If (IsCorrect And ((N < MIN\_N) Or (N > MAX\_N))) Then

Begin

Writeln('Значение не попадает в диапазон!');

IsCorrect := False;

End;

Until (IsCorrect);

SetLength(Coordinates, N, Length\_Of\_Arr);

For I := Low(Coordinates) To High(Coordinates) Do

Begin

Writeln('Введите ', I + 1,' пару координат (X, Y)');

Repeat

IsCorrect := True;

Write('Введите X [', MIN, '; ', MAX, ']: ');

Try

Readln(Coordinates[I,0]);

Except

Writeln('Проверьте корректность ввода данных!');

IsCorrect := False;

End;

If (IsCorrect And ((Coordinates[I,0] < MIN) Or (Coordinates[I,0] > MAX))) Then

Begin

Writeln('Значение не попадает в диапазон!');

IsCorrect := False;

End;

Until (IsCorrect);

Repeat

IsCorrect := True;

Write('Введите Y [', MIN, '; ', MAX, ']: ');

Try

Readln(Coordinates[I,1]);

Except

Writeln('Проверьте корректность ввода данных!');

IsCorrect := False;

End;

If (IsCorrect And ((Coordinates[I,1] < MIN) Or (Coordinates[I,1] > MAX))) Then

Begin

Writeln('Значение не попадает в диапазон!');

IsCorrect := False;

End;

Until (IsCorrect);

End;

SameCoordinate := False;

For I := Low(Coordinates) To High(Coordinates) Do

Begin

For J := I + 1 To High(Coordinates) Do

Begin

If (Coordinates[I,0] = Coordinates[J,0]) And ((Coordinates[I,1] = Coordinates[J,1])) Then

Begin

Writeln('Ошибка! Введенные координаты равны!');

SameCoordinate := True;

End;

End;

End;

For I := Low(Coordinates) To High(Coordinates) Do

Begin

For J := I + 2 To High(Coordinates) Do

Begin

Vector[1,1] := Coordinates[(I + 1) Mod N,0] - Coordinates[I Mod N,0];

Vector[1,2] := Coordinates[(I + 1) Mod N,1] - Coordinates[I Mod N,1];

Vector[2,1] := Coordinates[(J + 1) Mod N,0] - Coordinates[J Mod N,0];

Vector[2,2] := Coordinates[(J + 1) Mod N,1] - Coordinates[J Mod N,1];

VectorMultiplication := (Vector[1,1] \* Vector[2,2] - Vector[2,1] \* Vector[1,2]);

If VectorMultiplication <> 0 Then

Begin

X\_Test := ((Coordinates[I Mod N,0] \* Vector[1,2] \* Vector[2,1]) -

(Coordinates[J Mod N,0] \* Vector[2,2] \* Vector[1,1]) +

(Vector[1,1] \* Vector[2,1] \* (Coordinates[J Mod N,1]) -

Coordinates[I Mod N,1])) /

-VectorMultiplication;

If Coordinates[(I + 1) mod N, 0] - Coordinates[I mod N, 0] = 0 Then

Begin

If Coordinates[(I + 1) Mod N,0] > Coordinates[I Mod N,0] Then

Begin

Max\_X := Coordinates[(I + 1) Mod N,0];

Min\_X := Coordinates[I Mod N,0];

End

Else

Begin

Max\_X := Coordinates[I Mod N,0];

Min\_X := Coordinates[(I + 1) Mod N,0];

End;

If (X\_Test > Min\_X) And (X\_Test < Max\_X) Then

IsCorrectFigure := False;

End

Else

Begin

If Coordinates[(J + 1) Mod N,0] > Coordinates[J Mod N,0] Then

Begin

Max\_X := Coordinates[(J + 1) Mod N,0];

Min\_X := Coordinates[J Mod N,0];

End

Else

Begin

Max\_X := Coordinates[J Mod N,0];

Min\_X := Coordinates[(J + 1) Mod N,0];

End;

If (X\_Test > Min\_X) And (X\_Test < Max\_X) Then

IsCorrectFigure := False;

End;

End;

End;

End;

If (Not IsCorrectFigure) And (Not SameCoordinate) Then

Writeln('Ошибка! Стороны многоугольника пересекаются!');

If IsCorrectFigure Then

Begin

For I := Low(Coordinates) To High(Coordinates) Do

Begin

Vector[1,1] := Coordinates[(I + 1) Mod N,0] - Coordinates[I Mod N,0];

Vector[1,2] := Coordinates[(I + 1) Mod N,1] - Coordinates[I Mod N,1];

Vector[2,1] := Coordinates[(I + 2) Mod N,0] - Coordinates[(I + 1) Mod N,0];

Vector[2,2] := Coordinates[(I + 2) Mod N,1] - Coordinates[(I + 1) Mod N,1];

VectorMultiplication := (Vector[1,1] \* Vector[2,2] - Vector[2,1] \* Vector[1,2]);

If (VectorMultiplication \* PrevVectorMultiplication < 0) Then

NotConvexPolygon := False;

If VectorMultiplication = 0 Then

NotOnSameLine := False;

PrevVectorMultiplication := VectorMultiplication;

End;

End;

If (IsCorrectFigure And (Not SameCoordinate)) Then

Begin

If Not NotOnSameLine Then

Writeln('Ошибка! Стороны многоугольника лежат на одной прямой!')

Else If Not NotConvexPolygon Then

Writeln('Многоугольник не является выпуклым');

Max\_X := Coordinates[0,0];

Min\_X := Coordinates[0,0];

Max\_Y := Coordinates[0,1];

Min\_Y := Coordinates[0,1];

For I := Low(Coordinates) To High(Coordinates) Do

Begin

If Coordinates[I, 0] < Min\_X Then

Min\_X := Coordinates[I,0];

If Coordinates[I, 0] > Max\_X Then

Max\_X := Coordinates[I, 0];

If Coordinates[I, 1] < Min\_Y Then

Min\_Y := Coordinates[I, 1];

If Coordinates[I, 1] > Max\_Y Then

Max\_Y := Coordinates[I, 1];

End;

Mid\_X := (Min\_X + Max\_X) / 2;

Mid\_Y := (Min\_Y + Max\_Y) / 2;

For I := Low(Coordinates) To High(Coordinates) Do

Begin

Vector[1,1] := Coordinates[(I + 1) Mod N,0] - Mid\_X;

Vector[1,2] := Coordinates[(I + 1) Mod N,1] - Mid\_Y;

Vector[2,1] := Coordinates[(I + 2) Mod N,0] - Mid\_X;

Vector[2,2] := Coordinates[(I + 2) Mod N,1] - Mid\_Y;

VectorMultiplication := abs(Vector[1,1] \* Vector[2,2] - Vector[2,1] \* Vector[1,2]);

SquareOfFigure := SquareOfFigure + (0.5 \* (VectorMultiplication));

End;

If NotOnSameLine And NotConvexPolygon Then

Writeln('Площадь многоугольника = ', SquareOfFigure);

End;

Coordinates := Nil;

Readln;

End.

**Код программы С++:**

#include <iostream>

#include<cmath>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

const float MIN = -10000;

const float MAX = 10000;

const int MIN\_N = 3;

const int MAX\_N = 10000;

const int LENGTH\_OF\_ARR = 2;

int n = 0;

float vectorMultiplication;

float prevVectorMultiplication = 0;

float xTest;

float maxX;

float minX;

float maxY;

float minY;

float midX;

float midY;

float squareOfFigure = 0;

bool isCorrect = true;

bool notConvexPolygon = true;

bool notOnSameLine = true;

bool isCorrectFigure = true;

bool sameCoordinate = true;

cout << "Эта программа находит площадь N-угольника.\n";

do

{

isCorrect = false;

cout << "Введите количство вершин [" << MIN\_N << "; " << MAX\_N << "]:\n";

cin >> n;

if (cin.get() != '\n')

{

isCorrect = true;

cout << "Проверьте корректность ввода данных!\n";

cin.clear();

while (cin.get() != '\n');

}

if (!isCorrect && ((n < MIN\_N) || (n > MAX\_N)))

{

isCorrect = true;

cout << "Значение не попадает в диапазон!" << endl;

}

} while (isCorrect);

float \*\*coordinates = new float\* [n];

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

{

coordinates[i] = new float[LENGTH\_OF\_ARR];

}

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

{

cout << "Введите " << i + 1 << " пару координат(X, Y)\n";

do

{

isCorrect = false;

cout << "Введите X [" << MIN << "; " << MAX << "]:\n";

cin >> coordinates[i][0];

if (cin.get() != '\n')

{

isCorrect = true;

cout << "Проверьте корректность ввода данных!\n";

cin.clear();

while (cin.get() != '\n');

}

if (!isCorrect && ((coordinates[i][0] < MIN) || (coordinates[i][0] > MAX)))

{

isCorrect = true;

cout << "Значение не попадает в диапазон!" << endl;

}

} while (isCorrect);

do

{

isCorrect = false;

cout << "Введите Y [" << MIN << "; " << MAX << "]:\n";

cin >> coordinates[i][1];

if (cin.get() != '\n')

{

isCorrect = true;

cout << "Проверьте корректность ввода данных!\n";

cin.clear();

while (cin.get() != '\n');

}

if (!isCorrect && ((coordinates[i][1] < MIN) || (coordinates[i][1] > MAX)))

{

isCorrect = true;

cout << "Значение не попадает в диапазон!" << endl;

}

} while (isCorrect);

}

int i = 0;

int j = 0;

while ((i < n) && (sameCoordinate))

{

j = i + 1;

while ((j < n) && (sameCoordinate))

{

if ((coordinates[i][0] == coordinates[j][0]) && (coordinates[i][1] == coordinates[j][1]))

{

cout << "Ошибка! Введенные координаты равны!\n";

sameCoordinate = false;

}

j++;

}

i++;

}

float vec[2][2];

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

{

for (int j = i + 2; j < n; j++)

{

vec[0][0] = coordinates[(i + 1) % n][0] - coordinates[i % n][0];

vec[0][1] = coordinates[(i + 1) % n][1] - coordinates[i % n][1];

vec[1][0] = coordinates[(j + 1) % n][0] - coordinates[j % n][0];

vec[1][1] = coordinates[(j + 1) % n][1] - coordinates[j % n][1];

vectorMultiplication = vec[0][0] \* vec[1][1] - vec[1][0] \* vec[0][1];

if (vectorMultiplication != 0)

{

xTest = ((coordinates[i % n][0] \* vec[0][1] \* vec[1][0]) -

(coordinates[j % n][0] \* vec[1][1] \* vec[0][0]) +

(vec[1][1] \* vec[1][0] \* (coordinates[j % n][1]) -

coordinates[i % n][1])) / - vectorMultiplication;

if ((coordinates[(i + 1) % n][0] - coordinates[i % n][0]) == 0)

{

if (coordinates[(i + 1) % n][0] > coordinates[i % n][0])

{

maxX = coordinates[(i + 1) % n][0];

minX = coordinates[i % n][0];

}

else

{

minX = coordinates[(i + 1) % n][0];

maxX = coordinates[i % n][0];

}

if ((xTest > minX) && (xTest < maxX))

{

isCorrectFigure = false;

}

}

else

{

if (coordinates[(j + 1) % n][0] > coordinates[j % n][0])

{

maxX = coordinates[(j + 1) % n][0];

minX = coordinates[j % n][0];

}

else

{

minX = coordinates[(j + 1) % n][0];

maxX = coordinates[j % n][0];

}

if ((xTest > minX) && (xTest < maxX))

{

isCorrectFigure = false;

}

}

}

}

}

if ((!isCorrectFigure) && (!sameCoordinate))

{

cout << "Ошибка! Стороны многоугольника пересекаются!\n";

}

if (isCorrectFigure)

{

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

{

vec[0][0] = coordinates[(i + 1) % n][0] - coordinates[i % n][0];

vec[0][1] = coordinates[(i + 1) % n][1] - coordinates[i % n][1];

vec[1][0] = coordinates[(i + 2) % n][0] - coordinates[(i + 1) % n][0];

vec[1][1] = coordinates[(i + 2) % n][1] - coordinates[(i + 1) % n][1];

vectorMultiplication = vec[0][0] \* vec[1][1] - vec[1][0] \* vec[0][1];

if (vectorMultiplication \* prevVectorMultiplication < 0)

{

notConvexPolygon = false;

}

if (vectorMultiplication == 0)

{

notOnSameLine = false;

}

prevVectorMultiplication = vectorMultiplication;

}

}

if ((isCorrectFigure) && (!sameCoordinate))

{

if (!notOnSameLine)

{

cout << "Ошибка! Стороны многоугольника лежат на одной прямой!\n";

}

else if (!notConvexPolygon)

{

cout << "Многоугольник не является выпуклым";

}

maxX = coordinates[0][0];

minX = coordinates[0][0];

maxY = coordinates[0][1];

minY = coordinates[0][1];

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

{

if (coordinates[i][0] < minX)

{

minX = coordinates[i][0];

}

if (coordinates[i][0] > maxX)

{

maxX = coordinates[i][0];

}

if (coordinates[i][1] < minY)

{

minY = coordinates[i][1];

}

if (coordinates[i][1] > maxY)

{

maxY = coordinates[i][1];

}

}

midX = (minX + maxX) / 2;

midY = (minY + maxY) / 2;

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

{

vec[0][0] = coordinates[(i + 1) % n][0] - midX;

vec[0][1] = coordinates[(i + 1) % n][1] - midY;

vec[1][0] = coordinates[(i + 2) % n][0] - midX;

vec[1][1] = coordinates[(i + 2) % n][1] - midY;

vectorMultiplication = abs(vec[0][0] \* vec[1][1] - vec[1][0] \* vec[0][1]);

squareOfFigure = squareOfFigure + ((0.5) \* vectorMultiplication);

}

if (notOnSameLine && notConvexPolygon)

{

cout << "Площадь многоугольника = " << squareOfFigure << "\n\n";

}

}

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

{

delete[] coordinates[i];

}

delete[] coordinates;

return 0;

}

**Код программы Java:**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

final int MIN = -10000;

final int MAX = 10000;

final int MIN\_N = 3;

final int MAX\_N = 10000;

final int LENGTH\_OF\_ARR = 2;

int n = 0;

float vectorMultiplication;

float prevVectorMultiplication = 0;

float xTest;

float maxX;

float minX;

float maxY;

float minY;

float midX;

float midY;

double squareOfFigure = 0;

boolean isCorrect = true;

boolean notConvexPolygon = true;

boolean notOnSameLine = true;

boolean isCorrectFigure = true;

boolean sameCoordinate = true;

System.out.println("Эта программа находит площадь N-угольника:\n");

do {

isCorrect = false;

System.out.print("Введите количество вершин многоугольника [" + MIN\_N + "; " + MAX\_N + "]: ");

try {

n = Integer.parseInt(in.nextLine());

} catch (NumberFormatException e) {

System.out.print("Проверьте корректность ввода данных!\n");

isCorrect = true;

}

if ( !isCorrect && ((n < MIN\_N) || (n > MAX\_N))) {

System.out.print("Значение не попадает в диапазон!\n");

isCorrect = true;

}

} while (isCorrect);

float[][] coordinates = new float[n][LENGTH\_OF\_ARR];

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

System.out.println("Введите " + (i + 1) + " пару координат(X, Y)");

do {

isCorrect = false;

System.out.print("Введите X [" + MIN\_N + "; " + MAX\_N + "]: ");

try {

coordinates[i][0] = Float.parseFloat(in.nextLine());

} catch (NumberFormatException e) {

System.out.print("Проверьте корректность ввода данных!\n");

isCorrect = true;

}

if (!isCorrect && ((coordinates[i][0] < MIN) || (coordinates[i][0] > MAX))) {

System.out.print("Значение не попадает в диапазон!\n");

isCorrect = true;

}

} while (isCorrect);

do {

isCorrect = false;

System.out.print("Введите Y [" + MIN\_N + "; " + MAX\_N + "]: ");

try {

coordinates[i][1] = Float.parseFloat(in.nextLine());

} catch (NumberFormatException e) {

System.out.print("Проверьте корректность ввода данных!\n");

isCorrect = true;

}

if (!isCorrect && ((coordinates[i][1] < MIN) || (coordinates[i][1] > MAX))) {

System.out.print("Значение не попадает в диапазон!\n");

isCorrect = true;

}

} while (isCorrect);

}

int i = 0;

int j = 0;

while ((i < n) && (sameCoordinate)){

j = i + 1;

while ((j < n) && (sameCoordinate)){

if ((coordinates[i][0] == coordinates[j][0]) && (coordinates[i][1] == coordinates[j][1])){

System.out.println("Ошибка! Введенные координаты равны!\n");

sameCoordinate = false;

}

j++;

}

i++;

}

float[][] vec = new float[2][2];

for (i = 0; i < n; i++){

for (j = i + 2; j < n; j++){

vec[0][0] = coordinates[(i + 1) % n][0] - coordinates[i % n][0];

vec[0][1] = coordinates[(i + 1) % n][1] - coordinates[i % n][1];

vec[1][0] = coordinates[(j + 1) % n][0] - coordinates[j % n][0];

vec[1][1] = coordinates[(j + 1) % n][1] - coordinates[j % n][1];

vectorMultiplication = vec[0][0] \* vec[1][1] - vec[1][0] \* vec[0][1];

if (vectorMultiplication != 0){

xTest = ((coordinates[i % n][0] \* vec[0][1] \* vec[1][0]) -

(coordinates[j % n][0] \* vec[1][1] \* vec[0][0]) +

(vec[1][1] \* vec[1][0] \* (coordinates[j % n][1]) -

coordinates[i % n][1])) / - vectorMultiplication;

if ((coordinates[(i + 1) % n][0] - coordinates[i % n][0]) == 0){

if (coordinates[(i + 1) % n][0] > coordinates[i % n][0]){

maxX = coordinates[(i + 1) % n][0];

minX = coordinates[i % n][0];

}

else{

minX = coordinates[(i + 1) % n][0];

maxX = coordinates[i % n][0];

}

if ((xTest > minX) && (xTest < maxX)){

isCorrectFigure = false;

}

}

else{

if (coordinates[(j + 1) % n][0] > coordinates[j % n][0]){

maxX = coordinates[(j + 1) % n][0];

minX = coordinates[j % n][0];

}

else{

minX = coordinates[(j + 1) % n][0];

maxX = coordinates[j % n][0];

}

if ((xTest > minX) && (xTest < maxX)){

isCorrectFigure = false;

}

}

}

}

}

if ((!isCorrectFigure) && (!sameCoordinate)){

System.out.println("Ошибка! Стороны многоугольника пересекаются!");

}

if (isCorrectFigure){

for (i = 0; i < n; i++){

vec[0][0] = coordinates[(i + 1) % n][0] - coordinates[i % n][0];

vec[0][1] = coordinates[(i + 1) % n][1] - coordinates[i % n][1];

vec[1][0] = coordinates[(i + 2) % n][0] - coordinates[(i + 1) % n][0];

vec[1][1] = coordinates[(i + 2) % n][1] - coordinates[(i + 1) % n][1];

vectorMultiplication = vec[0][0] \* vec[1][1] - vec[1][0] \* vec[0][1];

if (vectorMultiplication \* prevVectorMultiplication < 0){

notConvexPolygon = false;

}

if (vectorMultiplication == 0){

notOnSameLine = false;

}

prevVectorMultiplication = vectorMultiplication;

}

}

if ((isCorrectFigure) && (!sameCoordinate)){

if (!notOnSameLine){

System.out.println("Ошибка! Стороны многоугольника лежат на одной прямой!\n");

}

else if (!notConvexPolygon){

System.out.println("Многоугольник не является выпуклым");

}

maxX = coordinates[0][0];

minX = coordinates[0][0];

maxY = coordinates[0][1];

minY = coordinates[0][1];

for (i = 0; i < n; i++){

if (coordinates[i][0] < minX){

minX = coordinates[i][0];

}

if (coordinates[i][0] > maxX){

maxX = coordinates[i][0];

}

if (coordinates[i][1] < minY){

minY = coordinates[i][1];

}

if (coordinates[i][1] > maxY){

maxY = coordinates[i][1];

}

}

midX = (minX + maxX) / 2;

midY = (minY + maxY) / 2;

for (i = 0; i < n; i++){

vec[0][0] = coordinates[(i + 1) % n][0] - midX;

vec[0][1] = coordinates[(i + 1) % n][1] - midY;

vec[1][0] = coordinates[(i + 2) % n][0] - midX;

vec[1][1] = coordinates[(i + 2) % n][1] - midY;

vectorMultiplication = Math.abs(vec[0][0] \* vec[1][1] - vec[1][0] \* vec[0][1]);

squareOfFigure = squareOfFigure + ((0.5) \* vectorMultiplication);

}

if (notOnSameLine && notConvexPolygon){

System.out.println("Площадь многоугольника = " + squareOfFigure);

}

}

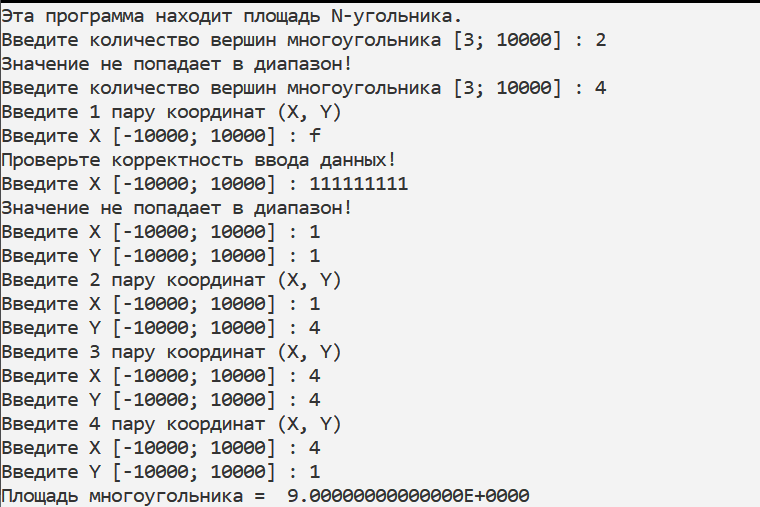
in.close();

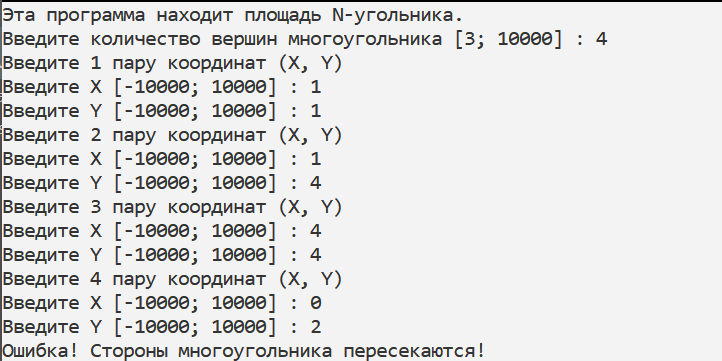
}

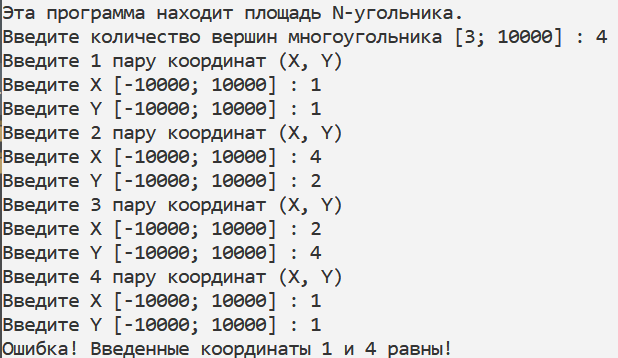
}

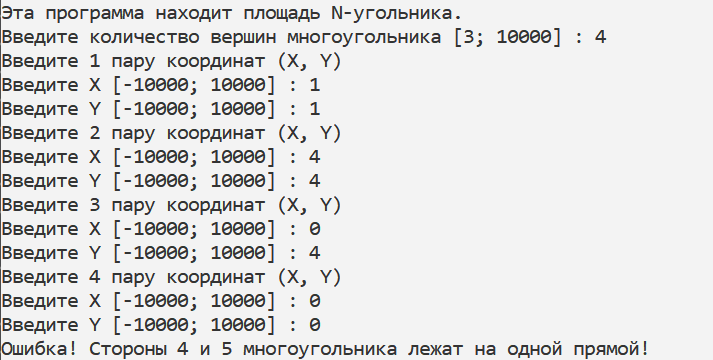
**Скриншоты:**

**Delphi:**

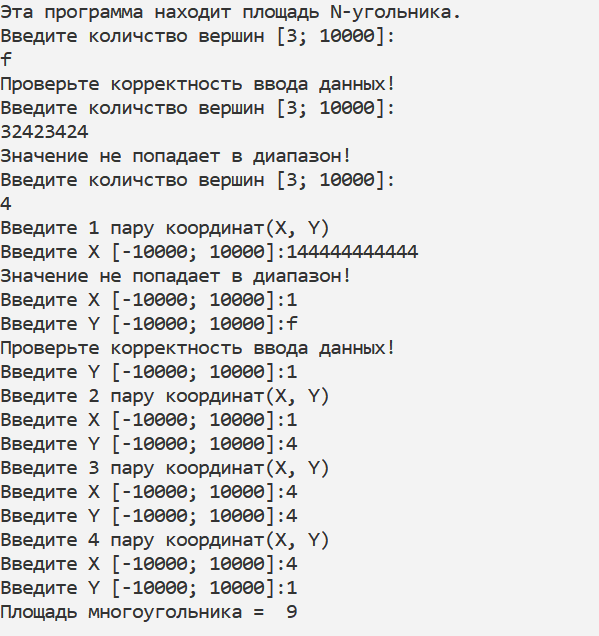
****

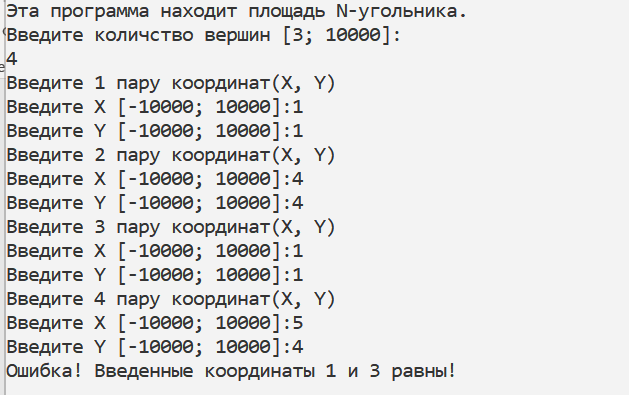
****

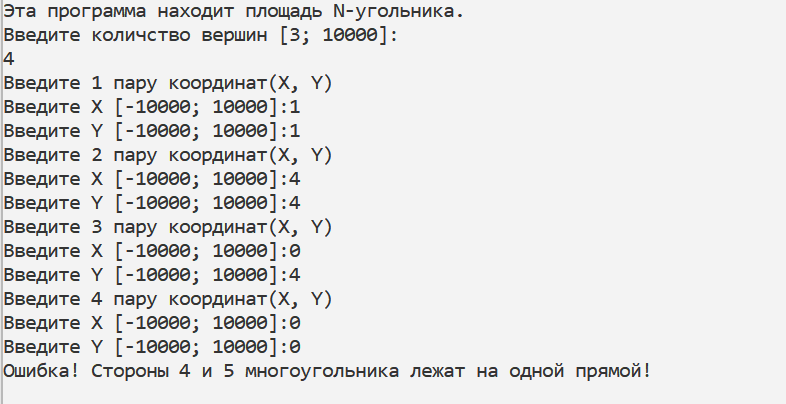


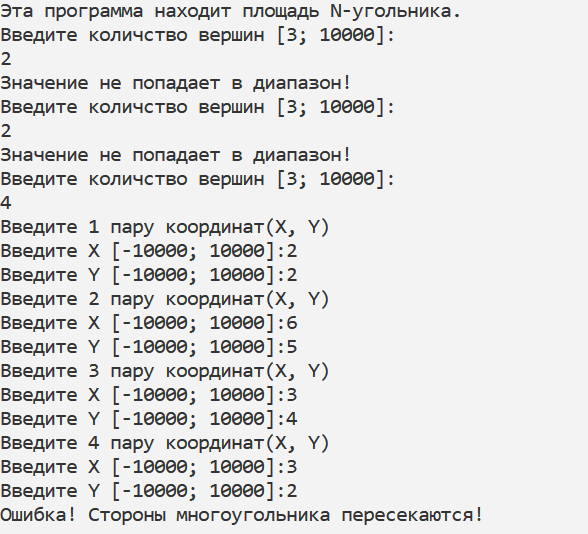
****

**C++:**

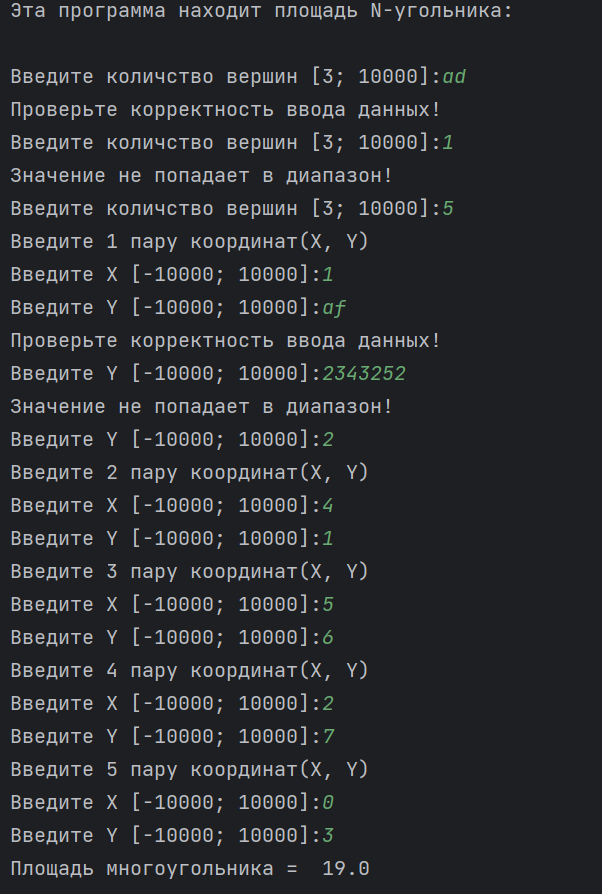
****

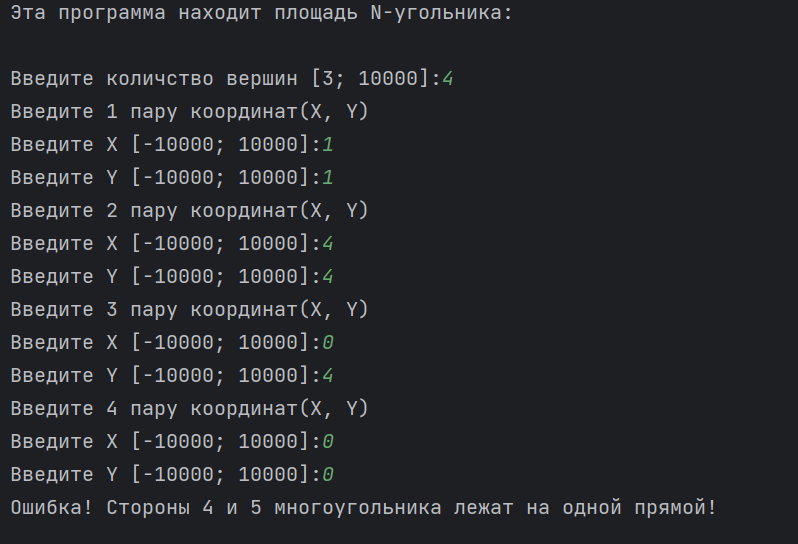
****

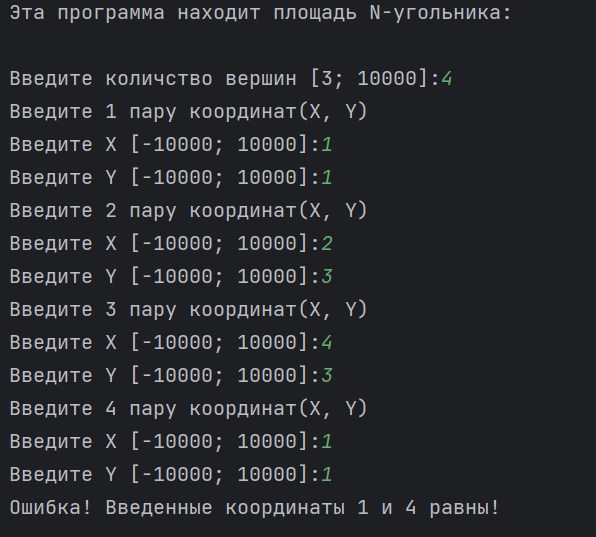
****

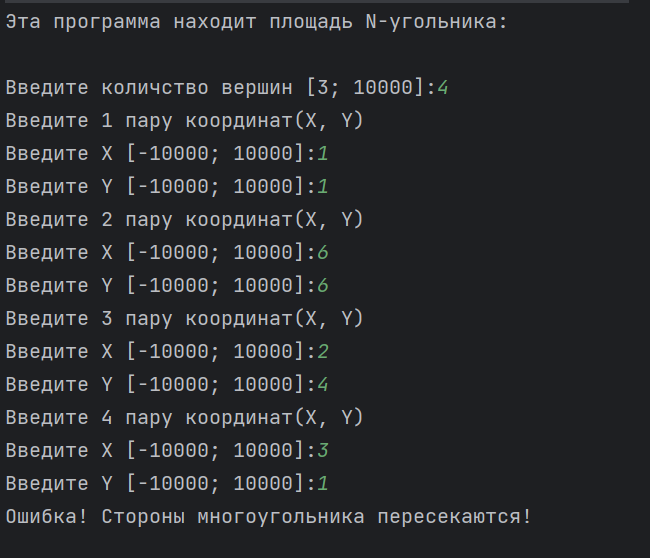
****

**Java:**

****

****

****

****

**Блок-схема:**





