Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Кафедра **«**Информационные технологии и автоматизированные системы**»**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №7.2**

Дисциплина: «Основы алгоритмизации и программирования»

Семестр 2

Тема: Строки.

Вариант 25.

Выполнил работу

Студент группы РИС-22-1Б

Поважный В. Е.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

г. Пермь-2023

**Введение**

Для выполнения лабораторной работы требуется сделать отчет программы. Создание программ – отличный способ практики программирования.

**Постановка задачи**

Написать функцию (или макроопределение), которая находит длину стороны по координатам его точек.. Написать функцию *square*, которая вычисляет площадь треугольника, заданного координатами вершин. Написать функцию square1 c переменным числом параметров, которая определяет площадь треугольника, содержащего диагональ наибольшей длины выпуклого многоугольника, заданного координатами своих вершин.

**Код программы**

#include <iostream>

using namespace std;

void square()

{

int ax, ay, bx, by, cx, cy;

cin >> ax >> ay >> bx >> by >> cx >> cy;

double p1 = sqrt((bx - ax) \* (bx - ax) + (by - ay) \* (by - ay));

double p2 = sqrt((bx - cx) \* (bx - cx) + (by - cy) \* (by - cy));

double p3 = sqrt((cx - ax) \* (cx - ax) + (cy - ay) \* (cy - ay));

double p = (p1 + p2 + p3) / 2;

cout << sqrt(p \* (p - p1) \* (p - p2) \* (p - p3)) << endl;

}

float square1(int k, ...)

{

int\* p = &k;

float length1 = sqrt(\*(p + 2) \* \*(p + 2) + \*(p + 4) \* \*(p + 4));

float length2 = sqrt(\*(p + 6) \* \*(p + 6) + \*(p + 8) \* \*(p + 8));

float length3 = sqrt(\*(p + 10) \* \*(p + 10) + \*(p + 12) \* \*(p + 12));

float buff = (length1 + length2 + length3) / 2;

return (sqrt(buff \* (buff - length1) \* (buff - length2) \* (buff - length3)));

}

int point\_vector(int arr\_x[], int arr\_y[], int arr\_point\_x[], int arr\_point\_y[], int n)

{

int sign = 1;

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

if (i == n - 1)

{

arr\_x[i] = arr\_point\_x[0] - arr\_point\_x[i];

arr\_y[i] = arr\_point\_y[0] - arr\_point\_y[i];

}

else

{

arr\_x[i] = arr\_point\_x[i] - arr\_point\_x[i + 1];

arr\_y[i] = arr\_point\_y[i] - arr\_point\_y[i + 1];

}

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

if (i == n - 1)

sign \*= arr\_x[i] \* arr\_y[0] - arr\_y[i] \* arr\_x[0];

else

sign \*= arr\_x[i] \* arr\_y[i + 1] - arr\_y[i] \* arr\_x[i + 1];

return sign;

}

int main()

{

int n = 0;

cin >> n;

int\* arr\_point\_x = new int[n];

int\* arr\_point\_y = new int[n];

int\* arr\_x = new int[n];

int\* arr\_y = new int[n];

float max = 0;

int i\_buff = 0;

int j\_buff = 0;

int buff1 = 0;

int buff2 = 0;

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

cin >> arr\_point\_y[i] >> arr\_point\_x[i];

}

if (point\_vector(arr\_x, arr\_y, arr\_point\_x, arr\_point\_y, n) < 0)

{

cout << "ERROR" << endl;

return 0;

}

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

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

{

int e = arr\_point\_x[i] - arr\_point\_x[j];

int r = arr\_point\_y[i] - arr\_point\_y[j];

double length = sqrt(e \* e + r \* r);

if (length > max)

{

max = length;

i\_buff = i;

j\_buff = j;

buff1 = e;

buff2 = r;

}

}

if (i\_buff == n - 1)

{

unsigned int p\_x1 = arr\_point\_x[i\_buff] - arr\_point\_x[0];

unsigned int p\_y1 = arr\_point\_y[i\_buff] - arr\_point\_y[0];

unsigned int p\_x2 = arr\_point\_x[0] - arr\_point\_x[j\_buff];

unsigned int p\_y2 = arr\_point\_y[0] - arr\_point\_y[j\_buff];

max = square1(6, p\_x1, p\_y1, p\_x2, p\_y2, buff1, buff2);

}

else

{

unsigned int p\_x1 = arr\_point\_x[i\_buff] - arr\_point\_x[i\_buff + 1];

unsigned int p\_y1 = arr\_point\_y[i\_buff] - arr\_point\_y[i\_buff + 1];

unsigned int p\_x2 = arr\_point\_x[i\_buff + 1] - arr\_point\_x[j\_buff];

unsigned int p\_y2 = arr\_point\_y[i\_buff + 1] - arr\_point\_y[j\_buff];

max = square1(6, p\_x1, p\_y1, p\_x2, p\_y2, buff1, buff2);

}

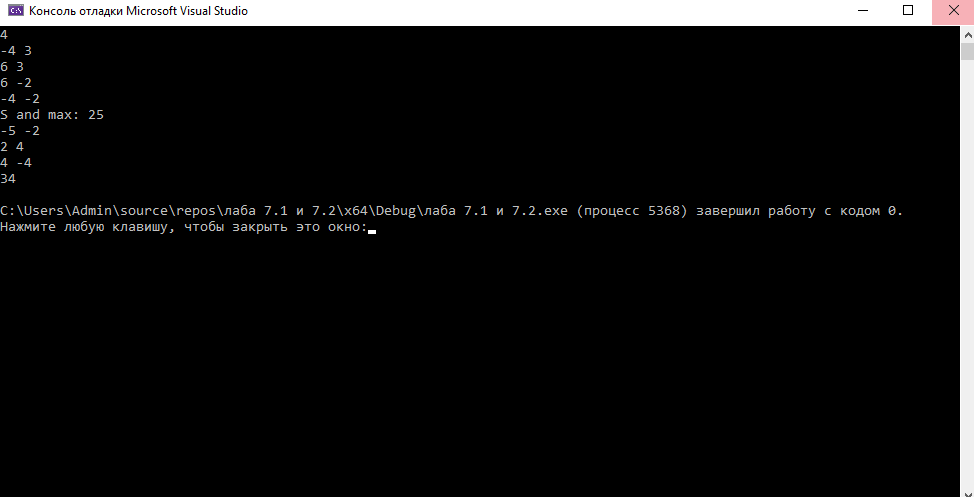
cout << "S and max: " << max << endl;

square();

return 0;

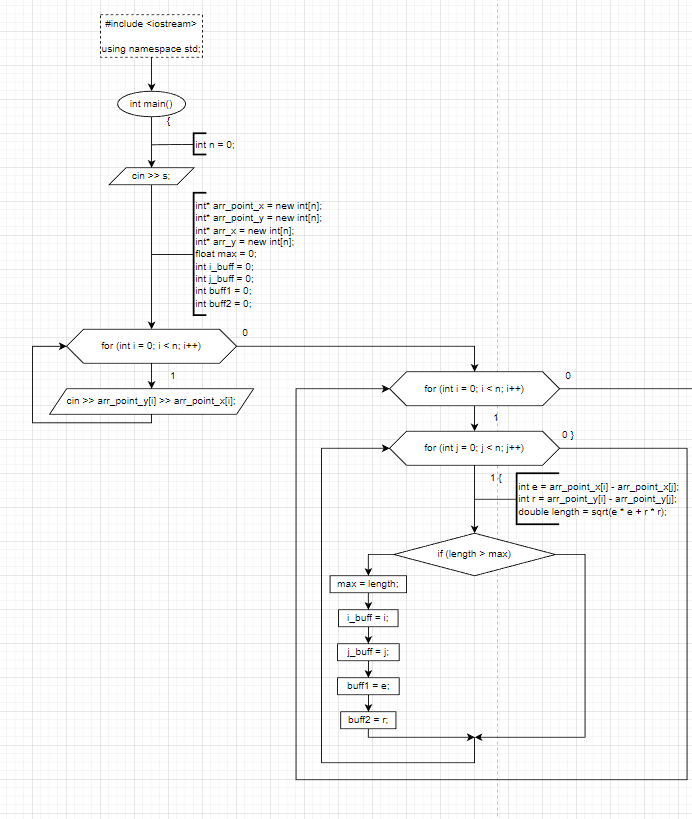
}

**Вывод программы**

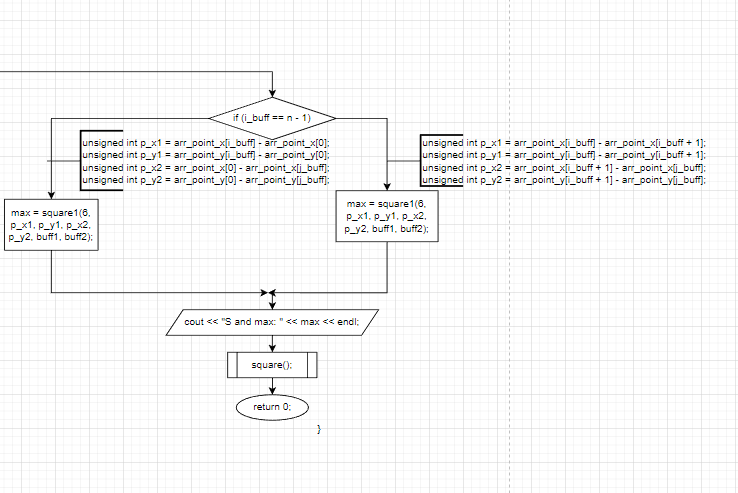
****

***Рисунок 1 – вывод программы***

**Блок-схема**

****

***Рисунок 2 – Схема алгоритма программы***

****

***Рисунок 3 – Продолжение схемы алгоритма программы***

**Вывод**

Программа выполняет свою задачу.

**Заключение**

Для решения задачи потребовались знания языка программирования, а конкретнее C++. Программа выполняет те условия, что были указаны в постановке задачи и работает без проблем.