

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение высшего  
образования  
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО  
ПРИБОРОСТРОЕНИЯ»  
(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Преподаватель

канд. техн. наук, доцент

Л.Н. Бариков

Отчет

по лабораторной работе № 7

по дисциплине Информатика

на тему: «перегрузка функций»

Работу выполнил

студент гр. 4342

С.М.Иванов

Санкт-Петербург

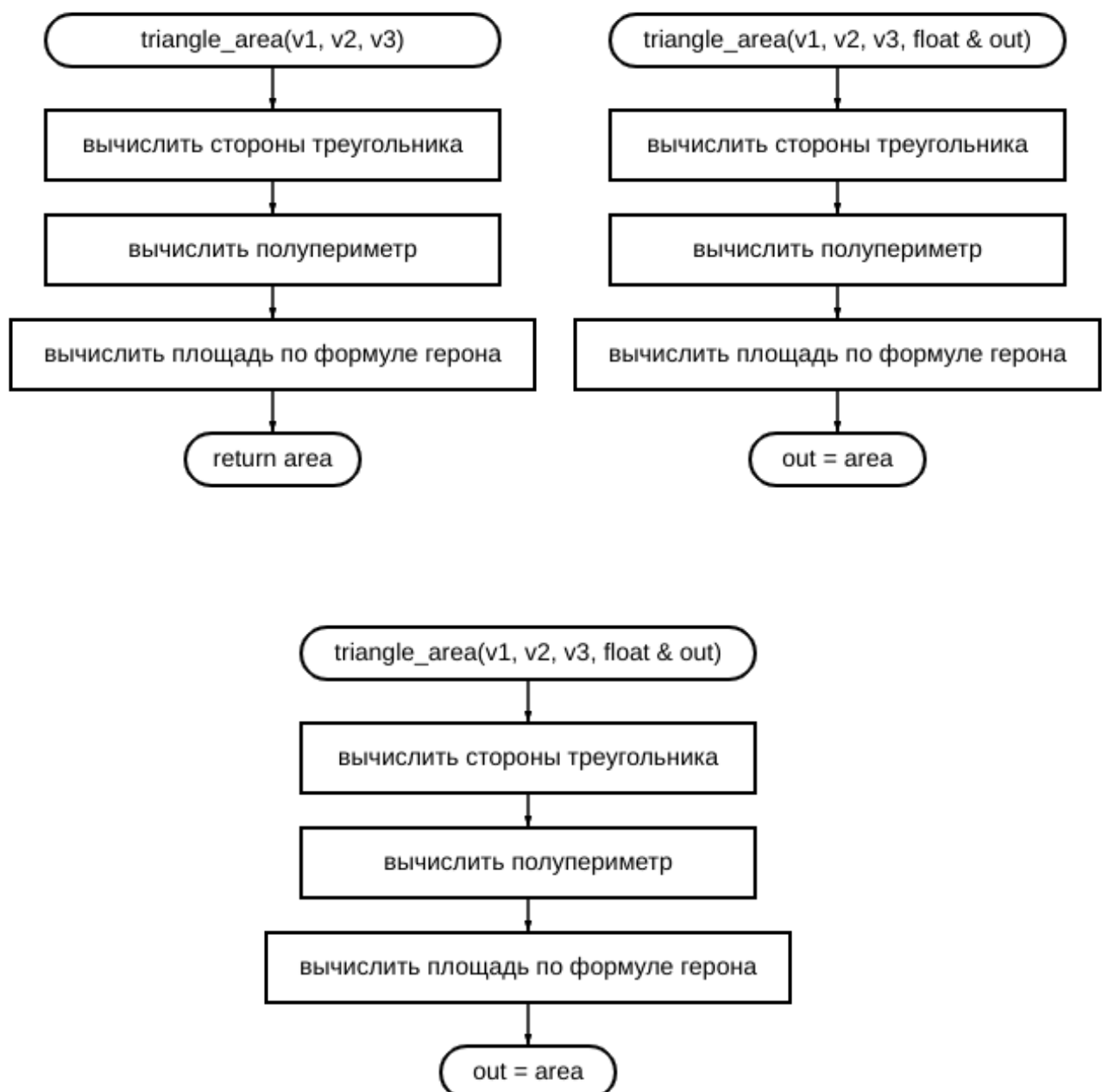
2024

## 16.

Разработать функцию, вычисляющую площадь треугольника. Ввести координаты вершин выпуклого четырёхугольника на плоскости. Используя разработанную функцию определить, являются ли стороны этого четырёхугольника попарно параллельными.

код на гитхабе для более удобного просмотра

[https://github.com/crimson-catfish/SUAI\\_labs/blob/main/lab7.cpp](https://github.com/crimson-catfish/SUAI_labs/blob/main/lab7.cpp)



Предоставленный алгоритм предназначен для определения того, является ли данный выпуклый четырехугольник параллелограммом или нет. Это делается путем вычисления площадей двух треугольников, образованных путем деления четырехугольника двумя его диагоналями. Если площади обеих пар противоположных треугольников равны, то алгоритм подтверждает, что данный четырехугольник является параллелограммом.

Алгоритм использует функцию `triangle_area` для вычисления площади треугольника. Эта функция принимает на вход координаты трех вершин треугольника и вычисляет площадь по формуле Герона. Функция перегружена для возврата области через возвращаемое значение, указатель или ссылку.

В основной функции алгоритм сначала считывает у пользователя координаты четырёх вершин четырёхугольника. Затем он использует функцию `triangle_area` для вычисления площадей треугольников и проверяет, равны ли площади противоположных треугольников. Этот процесс повторяется три раза, каждый раз используя другой способ возврата площади из функции `triangle_area` (через возвращаемое значение, указатель или ссылку).

На протяжении всего процесса программа выводит на консоль сообщения, указывающие, является ли данный четырехугольник параллелограммом или просто четырехугольником. После завершения проверок программа предлагает пользователю либо запустить алгоритм еще раз, либо выйти из программы.

```
#include <iostream>

#include <cmath>
using namespace std;

const float precision = 0.0001;

class Vertex
{
public: float x, y;
};

float triangle_area(Vertex v1, Vertex v2, Vertex v3)
{
    // Calculate the sides of the triangle
    float a = sqrt(pow(v2.x - v1.x, 2) + pow(v2.y - v1.y, 2));
    float b = sqrt(pow(v3.x - v2.x, 2) + pow(v3.y - v2.y, 2));
    float c = sqrt(pow(v3.x - v1.x, 2) + pow(v3.y - v1.y, 2));

    // Calculate the semi-perimeter of the triangle
    float s = (a + b + c) / 2;

    // Calculate the area of the triangle using Heron's formula
    float area = sqrt(s * (s - a) * (s - b) * (s - c));

    return area;
}

void triangle_area(Vertex v1, Vertex v2, Vertex v3, float * out)
{
    // Calculate the sides of the triangle
    float a = sqrt(pow(v2.x - v1.x, 2) + pow(v2.y - v1.y, 2));
    float b = sqrt(pow(v3.x - v2.x, 2) + pow(v3.y - v2.y, 2));
    float c = sqrt(pow(v3.x - v1.x, 2) + pow(v3.y - v1.y, 2));
```

```

// Calculate the semi-perimeter of the triangle
float s = (a + b + c) / 2;

// Calculate the area of the triangle using Heron's formula
float area = sqrt(s * (s - a) * (s - b) * (s - c));

*out = area;
}

void traingle_area(Vertex v1, Vertex v2, Vertex v3, float & out)
{
// Calculate the sides of the triangle
float a = sqrt(pow(v2.x - v1.x, 2) + pow(v2.y - v1.y, 2));
float b = sqrt(pow(v3.x - v2.x, 2) + pow(v3.y - v2.y, 2));
float c = sqrt(pow(v3.x - v1.x, 2) + pow(v3.y - v1.y, 2));

// Calculate the semi-perimeter of the triangle
float s = (a + b + c) / 2;

// Calculate the area of the triangle using Heron's formula
float area = sqrt(s * (s - a) * (s - b) * (s - c));

out = area;
}

bool compare_floats(float f1, float f2)
{
return fabs(f1 - f2) < precision;
}

int main()
{
char quitOrRestart = 'r';

do
{
Vertex quad[4];
cout << "Enter X and Y for each vertex of convex quadrilateral:\n";
for (int i = 0; i < 4; i++)
{
cin >> quad[i].x;
cin >> quad[i].y;
}
}

```

```
cout << "\nOutput via return: ";
if (compare_floats(triangle_area(quad[0], quad[1], quad[2]), triangle_area(quad[0],
quad[3], quad[2]))
&& compare_floats(triangle_area(quad[1], quad[2], quad[3]), triangle_area(quad[1],
quad[0], quad[3])))
{ cout << "This is a parallelogram!"; }
else cout << "This is just a quadrilateral";
```

```
float triangle_areas[4];
```

```
cout << "\nOutput via pointer: ";
triangle_area(quad[0], quad[1], quad[2], &triangle_areas[0]);
triangle_area(quad[0], quad[3], quad[2], &triangle_areas[1]);
triangle_area(quad[1], quad[2], quad[3], &triangle_areas[2]);
triangle_area(quad[1], quad[0], quad[3], &triangle_areas[3]);
if (compare_floats(triangle_areas[0], triangle_areas[1]) &&
compare_floats(triangle_areas[2], triangle_areas[3]))
{ cout << "This is a parallelogram!"; }
else cout << "This is just a quadrilateral";
```

```
cout << "\nOutput via reference: ";
triangle_area(quad[0], quad[1], quad[2], triangle_areas[0]);
triangle_area(quad[0], quad[3], quad[2], triangle_areas[1]);
triangle_area(quad[1], quad[2], quad[3], triangle_areas[2]);
triangle_area(quad[1], quad[0], quad[3], triangle_areas[3]);
if (compare_floats(triangle_areas[0], triangle_areas[1]) &&
compare_floats(triangle_areas[2], triangle_areas[3]))
{ cout << "This is a parallelogram!"; }
else cout << "This is just a quadrilateral";
```

```
cout << "\n\nEnter \"R\" to run again, or enter any other letter to quit programm: ";
cin >> quitOrRestart;
} while (quitOrRestart == 'r' || quitOrRestart == 'R');
```

```
return 0;
}
```