МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ЭЛЕКТРОТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА «ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АВТОМАТИЗИРОВАННЫХ СИСТЕМ»

ОТЧЁТ
«ЛАБОРАТОРНАЯ №7.2.»

Дисциплина: «Программирование»

Выполнил:
Студент группы ИВТ-21-2б
Безух Владимир Сергеевич

Проверил:
Доцент кафедры ИТАС
Полякова Ольга Андреевна

Пермь, 2022

Содержание

1. Постановка задачи

а) Написать функцию с неизвестным числом аргументом, которая последовательно находит расстояния от точки до точки для произвольного количества точек.

б) Написать функцию, которая вычисляет площадь треугольника, заданного координатами вершин.

в) Написать функцию, которая определяет площадь треугольника, содержащего диагональ наибольшей длины выпуклого многоугольника, заданного координатами вершин.

2. Анализ задачи

а) Через цикл последовательно находим расстояния от точки до точки. Для рационального решения задачи не нужно применять функцию с неизвестным числом аргументов. Достаточно использовать массив точек. Способ применён искусственно в учебных целях.

б) Для нахождения площади треугольника по координатам трёх вершин рационально применять векторное исчисление.

в) В выпуклом n-угольнике, начиная с многоугольника с 4 сторонами, можно проводить диагонали — линии, соединяющие несмежные вершины. Не все диагонали образуют треугольники.

Пронумеруем все вершины многоугольника от 0 до (n – 1) в порядке последовательного обхода против часовой стрелки. Для того, чтобы диагональ выпуклого n-угольника образовывала треугольник, необходимо соединить две вершины между которыми в порядке последовательного обхода будет только одна вершина (рис .1). Количество таких диагоналей равно (n – 2). Решу задачу рационально без искусственных приёмов.

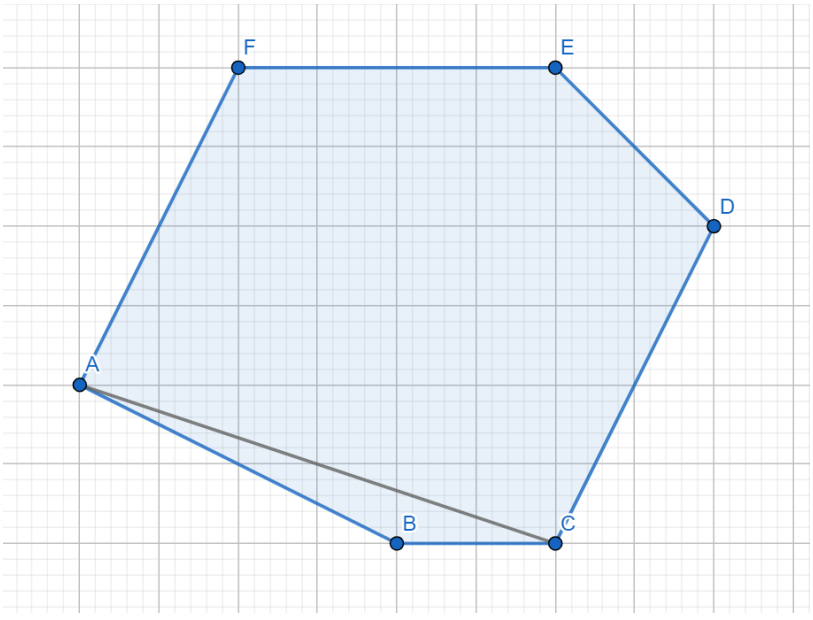


Рисунок 1 — Выпуклый многоугольник с диагональю, образующей треугольник

3. Описание переменных

struct Point { private: double x, y; … } — Структура данных для удобной работы с координатами точки.

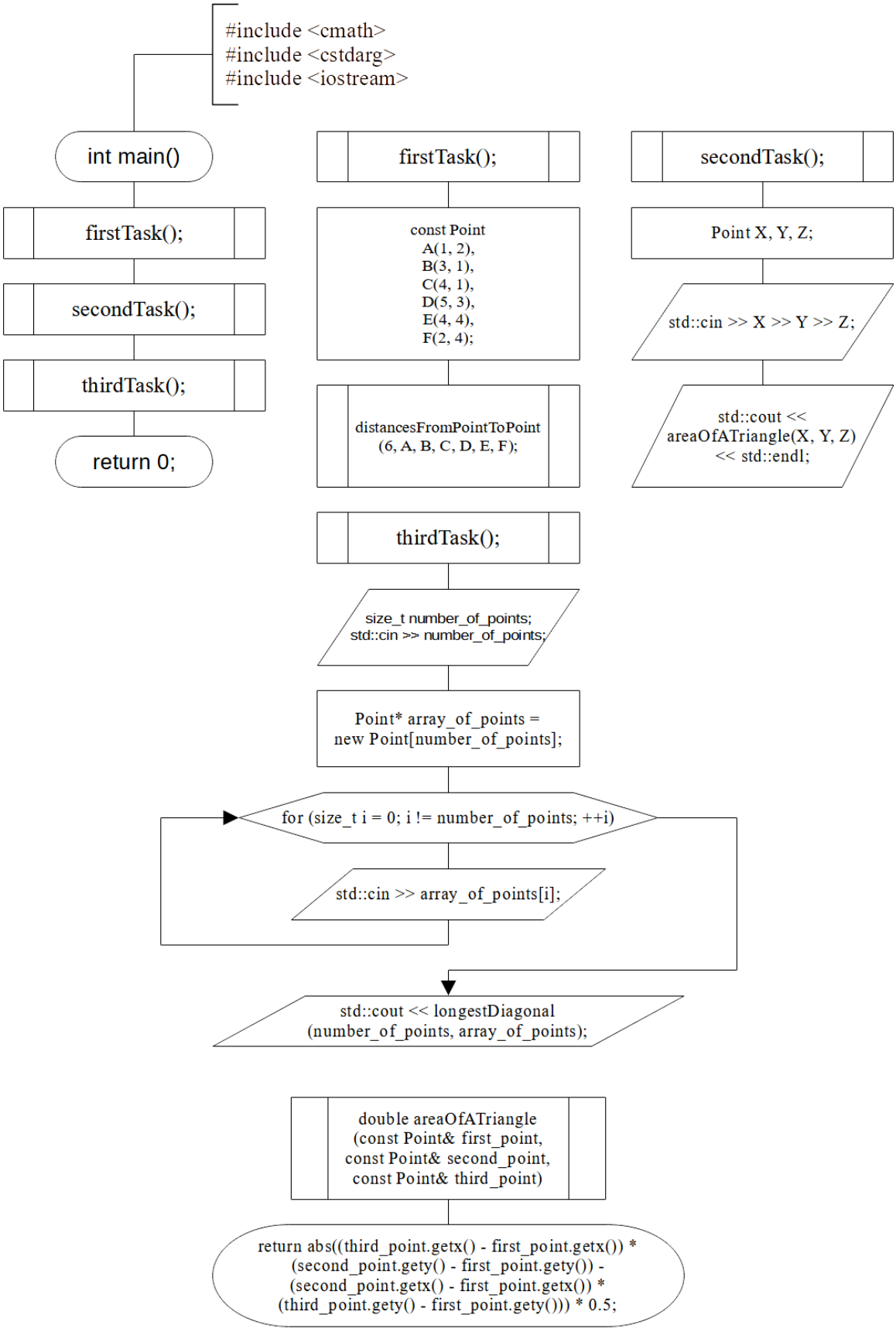
Point X, Y, Z; cin >> X >> Y >> Z; — Точки для демонстрации работы функции по нахождению площади треугольника.

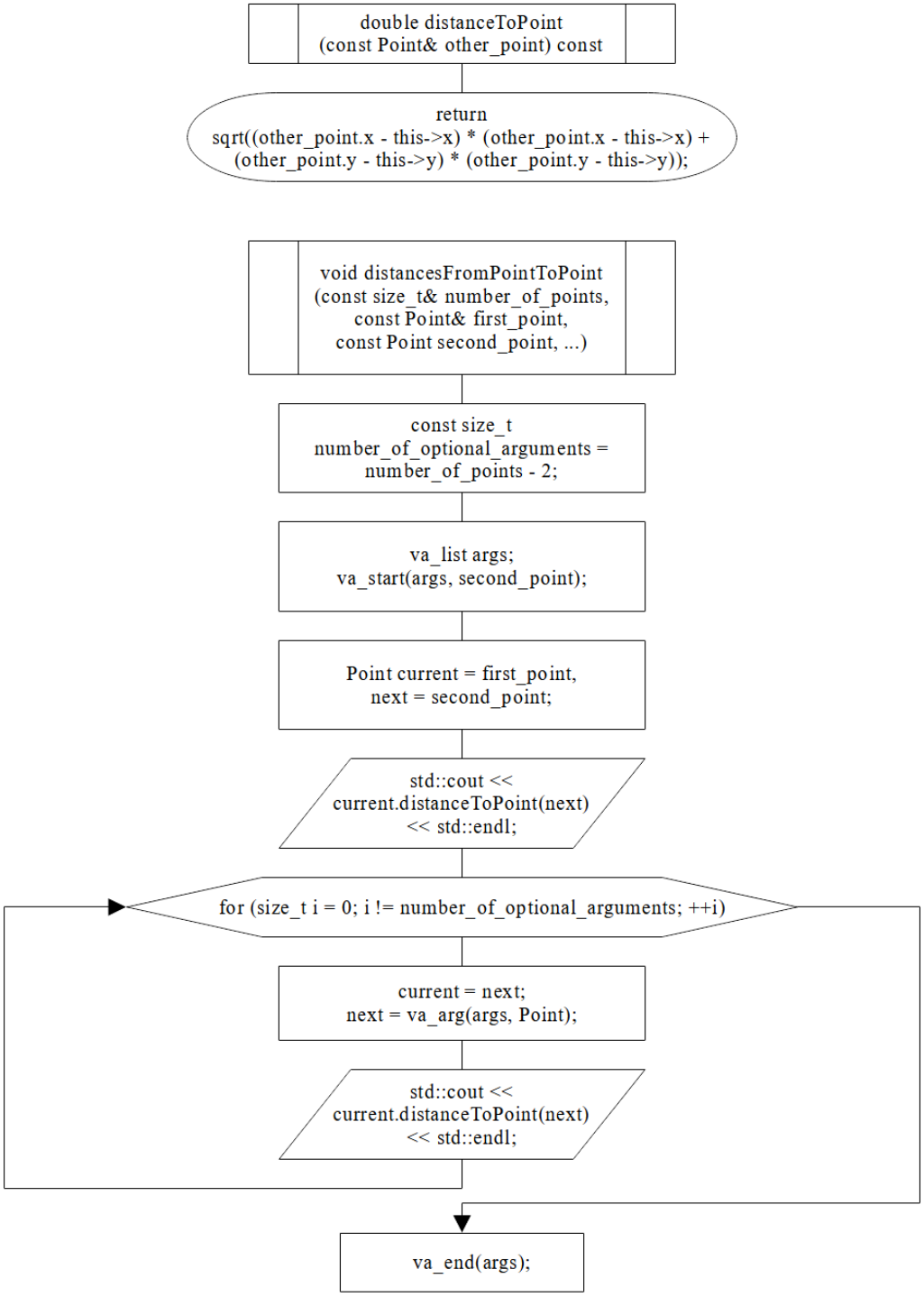
const Point A(1, 2), B(3, 1), C(4, 1), D(5, 3), E(4, 4), F(2, 4); — Точки для демонстрации работы функции по последовательному нахождению расстояния от точки до точки.

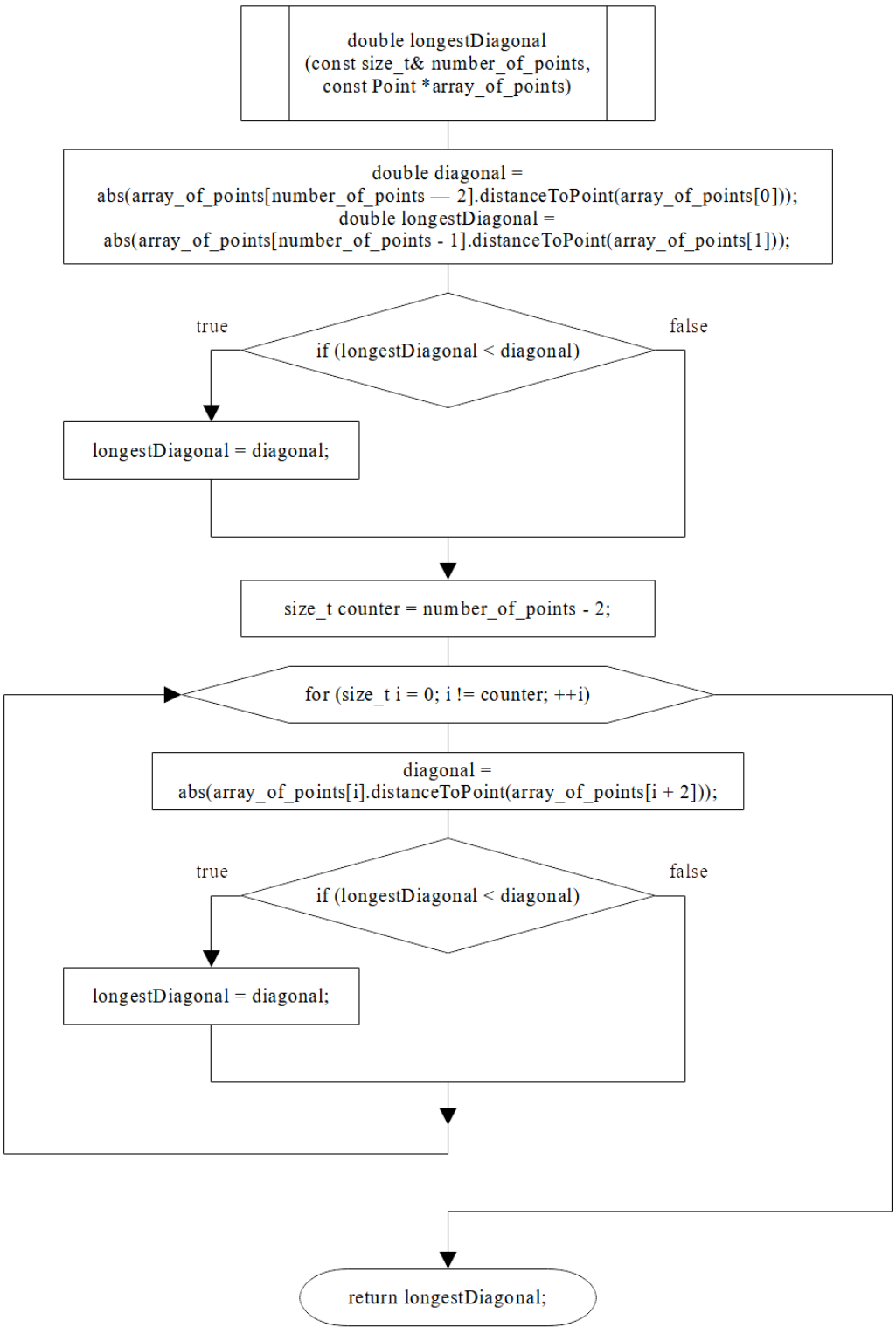
size_t number_of_points; cin >> number_of_points;

Point* array_of_points = new Point[number_of_points]; — Массив точек для демонстрации работы функции по нахождению самой длинной диагонали, образующей треугольник в многоугольнике.

4. Блок-схемы


#include <cmath>
#include <cstdarg>
#include <iostream>
int main()
firstTask();
secondTask();
thirdTask();
return 0;
firstTask();
const Point
A(1, 2),
B(3, 1),
C(4, 1),
D(5, 3),
E(4, 4),
F(2, 4);
distancesFromPointToPoint
(6, A, B, C, D, E, F);
secondTask();
Point X, Y, Z;
std::cin >> X >> Y >> Z;
std::cout <<
areaOfATriangle(X, Y, Z)
<< std::endl;
thirdTask();
size_t number_of_points;
std::cin >> number_of_points;
Point* array_of_points =
new Point[number_of_points];
for (size_t i = 0; i != number_of_points; ++i)
std::cin >> array_of_points[i];
std::cout << longestDiagonal
(number_of_points, array_of_points);
double areaOfATriangle
(const Point& first_point,
const Point& second_point,
const Point& third_point)
return abs((third_point.getx() - first_point.getx()) *
(second_point.gety() - first_point.gety()) -
(second_point.getx() - first_point.getx()) *
(third_point.gety() - first_point.gety())) * 0.5;


double distanceToPoint
(const Point& other_point) const
return
sqrt((other_point.x - this->x) * (other_point.x - this->x) +
(other_point.y - this->y) * (other_point.y - this->y));
void distancesFromPointToPoint
(const size_t& number_of_points,
const Point& first_point,
const Point second_point, ...)
const size_t
number_of_optional_arguments =
number_of_points - 2;
va_list args;
va_start(args, second_point);
Point current = first_point,
next = second_point;
std::cout <<
current.distanceToPoint(next)
<< std::endl;
for (size_t i = 0; i != number_of_optional_arguments; ++i)
current = next;
next = va_arg(args, Point);
std::cout <<
current.distanceToPoint(next)
<< std::endl;
va_end(args);


double longestDiagonal
(const size_t& number_of_points,
const Point *array_of_points)
double diagonal =
abs(array_of_points[number_of_points — 2].distanceToPoint(array_of_points[0]));
double longestDiagonal =
abs(array_of_points[number_of_points - 1].distanceToPoint(array_of_points[1]));
true
if (longestDiagonal < diagonal)
false
longestDiagonal = diagonal;
size_t counter = number_of_points - 2;
for (size_t i = 0; i != counter; ++i)
diagonal =
abs(array_of_points[i].distanceToPoint(array_of_points[i + 2]));
true
if (longestDiagonal < diagonal)
false
longestDiagonal = diagonal;
return longestDiagonal;

5. Исходный код

```
#include <cmath>
#include <cstdarg>
#include <iostream>

struct Point
{
private:
    double x, y;

public:
    void setx(double x) { this->x = x; }
    double getx() const { return this->x; }

    void sety(double y) { this->y = y; }
    double gety() const { return this->y; }

    Point() : x(0.0), y(0.0) {};
    Point(double x, double y) : x(x), y(y) {};

    double distanceToPoint(const Point& other_point) const {
        return sqrt((other_point.x - this->x) * (other_point.x - this->x) +
                    (other_point.y - this->y) * (other_point.y - this->y));
    }

    Point& operator=(const Point& right) {
        this->x = right.x;
        this->y = right.y;
        return *this;
    }

    friend std::istream& operator>>(std::istream& cin_, Point& enter) {
        cin_ >> enter.x;
        cin_ >> enter.y;
        return cin_;
    }
};

double areaOfATriangle(const Point& first_point,
                       const Point& second_point,
                       const Point& third_point)
{
    return abs((third_point.getx() - first_point.getx()) *
               (second_point.gety() - first_point.gety()) -
               (second_point.getx() - first_point.getx()) *
               (third_point.gety() - first_point.gety())) * 0.5;
}
```

```
void distancesFromPointToPoint(const size_t& number_of_points,
                               const Point& first_point,
                               const Point second_point, ...)
{
    const size_t number_of_optional_arguments = number_of_points - 2;

    va_list args;
    va_start(args, second_point);

    Point current = first_point, next = second_point;
    std::cout << current.distanceToPoint(next) << std::endl;

    for (size_t i = 0; i != number_of_optional_arguments; ++i)
    {
        current = next;
        next = va_arg(args, Point);
        std::cout << current.distanceToPoint(next) << std::endl;
    }

    va_end(args);
}

double longestDiagonal(const size_t& number_of_points,
                       const Point *array_of_points)
{
    double diagonal =
abs(array_of_points[number_of_points - 2].distanceToPoint(array_of_points[0]));
    double longestDiagonal =
abs(array_of_points[number_of_points - 1].distanceToPoint(array_of_points[1]));

    if (longestDiagonal < diagonal)
        longestDiagonal = diagonal;

    size_t counter = number_of_points - 2;

    for (size_t i = 0; i != counter; ++i)
    {
        diagonal = abs(array_of_points[i].distanceToPoint(array_of_points[i + 2]));

        if (longestDiagonal < diagonal)
            longestDiagonal = diagonal;
    }

    return longestDiagonal;
}

void firstTask()
{
    const Point A(1, 2), B(3, 1), C(4, 1), D(5, 3), E(4, 4), F(2, 4);
    distancesFromPointToPoint(6, A, B, C, D, E, F);
}

void secondTask()
{
    Point X, Y, Z; std::cin >> X >> Y >> Z;
    std::cout << areaOfATriangle(X, Y, Z) << std::endl;
}
```

```
void thirdTask()
{
    size_t number_of_points; std::cin >> number_of_points;
    Point* array_of_points = new Point[number_of_points];
    for (size_t i = 0; i != number_of_points; ++i)
        std::cin >> array_of_points[i];

    std::cout << longestDiagonal(number_of_points, array_of_points);
}


int main()
{
    firstTask();
    secondTask();
    thirdTask();
}
```

6. Консольный интерфейс программы

```
void firstTask()
{
    const Point A(1, 2), B(3, 1), C(4, 1), D(5, 3), E(4, 4), F(2, 4);
    distancesFromPointToPoint(6, A, B, C, D, E, F);
}

void secondTask()
{
    Point X, Y, Z; std::cin >> X >> Y >> Z;
    std::cout << areaOfATriangle(X, Y, Z) << std::endl;
}

void thirdTask()
{
    size_t number_of_points; std::cin >> number_of_points;
    Point* array_of_points = new Point[number_of_points];
    for (size_t i = 0; i != number_of_points; ++i)
        std::cin >> array_of_points[i];

    std::cout << longestDiagonal(number_of_points, array_of_points);
}
```

```
Консоль отладки М
2.23607
1
2.23607
1.41421
2
0 0
5 0
0 5
12.5
6
1 2
3 1
4 1
5 3
4 4
2 4
3.60555
D:\Учебные папки\П
с 10488) завершил
Чтобы автоматическ
томатически закрыт
Нажмите любую клав
```

7. Анализ результатов

а) Последовательно в цикле вычисляем расстояния от точки до точки с помощью функции-члена структуры Point.

б) Находим площадь треугольника через векторное исчисление.

в) Находим длины диагоналей как расстояния между нужными вершинами.