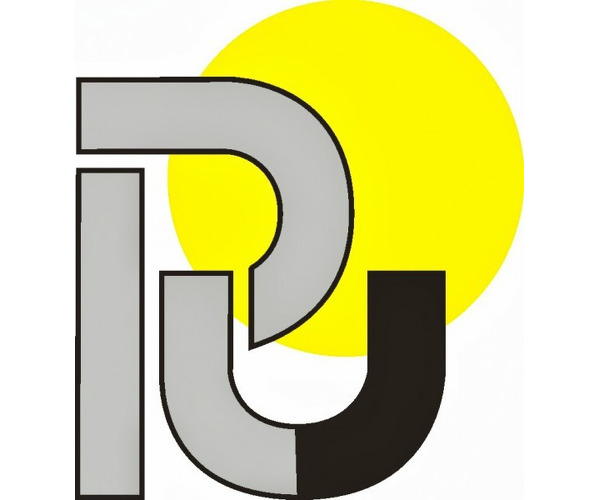
*РУ „Ангел Кънчев“ – Русе*

*Факултет природни науки и образование*

******

Курсова задача по Проектиране и анализ на алгоритми

на

Никола Кирилов Кирилов

Фак №: 216265

От 40а група

Дата: …………… Проверил:…………..

Съдържание

[Условие на задачата: 3](#_Toc148472125)

[Обяснение на решението: 4](#_Toc148472126)

[Код на задачата: 5](#_Toc148472127)

[Примери: 6](#_Toc148472128)

# Условие на задачата:

﻿ **Двойствена начупена линия**

Зададена е начупена линия P. Двойствена на P ще ннаричаме друга начупена линия, върховете на която съвпадат със средите на отсечките, съставящи P. Ще означим двойствената начупена линия на P от ред k с dual(P, k) и ще я определим по следния начин:

* Двойствената начупена линия линия от нулев ред е самата P (dual(P, 0) P);
* Ако k > 0, то двойствената начупена линия от ред k е dual(P, k -1).

Напишете програма, която, за зададена начупена линия P и число k, namira dylvinata na dual(P, k).

**Вход**

От първия ред на стандартния вход се въвежда цяло число n (3 n 100). От всеки от следващите n редове се въвеждат по две цели числа и – координати на i – тия връх на начупената линия. Всички и не надвишават по абсолютна стойност. Последният ред на входа съдържа едно цяло число k (0 < k < 10).

**Изход**

На един ред на стандартния изход изведете едно число – пресметнатата дължина с точност или по-голяма.

# Обяснение на решението:

Първо ще приемем от колко върхове се състои начупената линия, след което с помощта на цикъл ще ги запишем в списъка – points. След това от списъка с точките взимаме началните и крайните координати на всички линии/вектори и ги записваме в списък – vectors като обекти от тип Vector, като към типа Vector имаме предварително инициализирани функции calculateMidCords(), която намира координатите (x, y) за съответния вектор по формулите:

* x = (+ ) / 2
* y = (+ ) / 2

и функцията length() която изчислява дължината на вектора по формулата:

Реалното намиране на дължината на линните от k – ти ред се случва във функцията calculateLengthOfNth(list<Vector>, int), която като параметри приема лист от вектори и число което определя до кой ред да търси. Ако програмат получи 0 за търсен ред тя директно минава през всяка линия и изчислява сумата от резултатите на функцията length(). Ако k е по голямо от 0 функцията рекурсивно намира средите на всички линии от 0 до k - тия ред и накрая връща сумата от дължините както е описано по – горе, като за крайно условие за излизане от рекурсията се приема когато k = 0 тъй като като аргумент и подаваме (vectors , k-1).

# Код на задачата:

#include <iostream>

#include <iomanip>

#include <list>

#include <math.h>

using namespace std;

struct Point {

private:

float x, y;

public:

Point()

{

this->x = 0;

this->y = 0;

}

Point(float x, float y) {

this->x = x;

this->y = y;

}

float getX()

{

return x;

}

float getY()

{

return y;

}

};

struct Vector {

private:

Point start;

Point end;

public:

Vector()

{

this->start = Point(0, 0);

this->end = Point(1, 1);

};

Vector(float x1, float y1, float x2, float y2) {

this->start = Point(x1, y1);

this->end = Point(x2, y2);

}

Vector(Point start, Point end)

{

this->start = start;

this->end = end;

}

Vector(Vector vec1, Vector vec2)

{

this->start = vec1.calculateMidCords();

this->end = vec2.calculateMidCords();

}

Point getStart()

{

return this->start;

}

Point getEnd()

{

return this->end;

}

Point calculateMidCords()

{

Point midPoint = Point(

((this->start.getX() + this->end.getX()) / 2),

((this->start.getY() + this->end.getY()) / 2)

);

return midPoint;

}

float length()

{

float length = 0;

return length = sqrt(

((end.getX() - start.getX()) \* (end.getX() - start.getX()))

+

((end.getY() - start.getY()) \* (end.getY() - start.getY())));

}

};

list<Point> enterPoints(int count)

{

list<Point> points;

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

{

float x, y;

cin >> x >> y;

Point point = Point(x, y);

points.push\_back(point);

}

return points;

}

list<Vector> addVectors(list<Point> points)

{

list<Vector> vectors; int pointSize = points.size();

Vector connector = Vector(points.back(), points.front());

for (int i = 0; i < pointSize - 1; i++)

{

Point firstPoint = points.front();

points.pop\_front();

Vector vector = Vector(

firstPoint.getX(), firstPoint.getY(),

points.front().getX(), points.front().getY());

vectors.push\_back(vector);

}

vectors.push\_back(connector);

return vectors;

}

list<Vector> findMidVectors(list<Vector> vectors)

{

list<Vector> midVectors; int vectorsSize = vectors.size();

Vector connector = Vector(vectors.back().calculateMidCords(), vectors.front().calculateMidCords());

for (int i = 0; i < vectorsSize - 1; i++)

{

Point start = vectors.front().calculateMidCords();

vectors.pop\_front();

Point end = vectors.front().calculateMidCords();

midVectors.push\_back(Vector(start, end));

}

midVectors.push\_back(connector);

return midVectors;

}

float calculateLengthOfNth(list<Vector> midVectors, int row)

{

float length = 0;

if (row > 1)

{

list<Vector> innerVectors;

int midVecSize = midVectors.size();

Vector connector = Vector(midVectors.back(), midVectors.front());

for (int i = 0; i < midVecSize - 1; i++)

{

Vector Vec1 = midVectors.front();

midVectors.pop\_front();

Vector Vec2 = Vector(Vec1, midVectors.front());

innerVectors.push\_back(Vec2);

}

innerVectors.push\_back(connector);

length += calculateLengthOfNth(innerVectors, row - 1);

}

else

{

{

int midVecSize = midVectors.size();

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

{

length += midVectors.front().length();

midVectors.pop\_front();

}

}

}

return length;

}

int main()

{

int pointsCount;

cin >> pointsCount;

list<Point> points = enterPoints(pointsCount);

list<Vector> vectors = addVectors(points);

list<Vector> midVectors = findMidVectors(vectors);

int row;

cin >> row;

float length = calculateLengthOfNth(vectors, row);

if (row == 0) length = calculateLengthOfNth(vectors, row);

else length = calculateLengthOfNth(midVectors, row);

cout << fixed << setprecision(10) << length << endl;

# }

# Примери:

1. Вход:

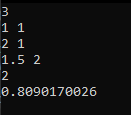
3

1 1

2 1

1.5 2

2  
Изход:

0.8090170026

1. Вход:

4

0 0

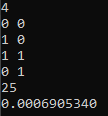
1 0

1 1

0 1

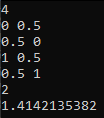
25

Изход:

0.0006905340

1. Вход:

Изход:

1.4142135382

1. Вход:

5

1 1

2 1

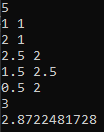
2.5 2

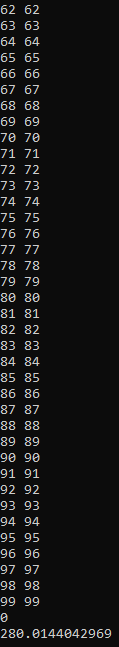
1.5 2.5

0.5 2

3

Изход:

2.8722481728

1. Вход:

100

0 0

1 1

2 2

3 3

… …

99 99

0

Изход:

280.0144042969