Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ОТЧЕТ

по разминочной работе №1

Тема работы: Поиск минимального элемента

Выполнил

студент: гр. 551004 Довыдёнок М.А.

Проверил: Фадеева Е.П.

Минск 2016

Содержание

[1 Постановка задачи 3](#_Toc444581649)

[2 Описание алгоритмов 4](#_Toc444581650)

[3 Основные расчетные формулы 5](#_Toc444581651)

[4 Результаты расчетов и тестирование программы 6](#_Toc444581652)

[Приложение А 7](#_Toc444581653)

# Постановка задачи

На плоскости расположены 10 точек, которые заданы своими координатами, найти на оси абсцисс точку, максимальное расстояние от которой до точек было бы минимальным

На плоскости расположены N точек, которые заданы своими координатами, найти на оси абсцисс точку, максимальное расстояние от которой до точек было бы минимальным

Вывести координаты найденной точки и максимальное расстояние от неё до точек

# Описание алгоритмов

Таблица 1 – Описание алгоритмов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование алгоритма | Назначение алгоритма | Формальные параметры | Рекомендуемый тип |
| 1. | getEps(result) | Запрашивает ввод и возвращает result | result – real Возвращаемый параметр: result | Функция. result – результат функции |
| 2 | getPoints(result) | Запрашивает ввод количества N и координат точек.  Возвращает их в result | result – array [0..n-1] of TPoint. TPoint – запись из двух полей. Возвращаемый параметр: result | Функция. result – результат функции |
| 3 | getDistance(point1, point2, result) | Вычисляет расстояние result между двумя точками | result – real. point1, point2 - TPoint. Возвращаемый параметр: result | Функция. result – результат функции |
| 4 | getSegment(points, min, max) | Находит проверяемый отрезок и возвращающает его координаты в min и max | points – array [0..n-1] of TPoint. min, max - TPoint  Возвращаемые параметры: min, max | Процедура |
| 5 | getMaxOfDistances (points, point, result) | Находит наибольшие расстояния result до точек справа и слева от текущей точки point | result – TDistances. TDistances – запись из 2 полей. points – array [0..n-1] of TPoint. point - TPoint.  Возвращаемый параметр:  result | Функция. result – результат функции |
| 6 | getOutputLength(eps, result) | Вычисляет длину поля вывода result для заданного eps | result, eps.  Возвращаемый параметр:  result | Функция. result – результат функции |
| 7 | outputResult (point, distance, outputLength) | Выводит точку point и расстояние distance, с outputLength знаками после запятой | point - TPoint, distance - TDistance, outputLength - Integer | Процедура |
| 8 | binarySearch (points, currentPoint, pointLeft, pointRight, currentDistances, eps) | Реализация алгоритма бинарного поиска, на отрезке pointLeft - pointRight | points – array [0..n-1] of TPoint. currentPoint, pointLeft, pointRight - TPoint, currentDistances - TDistances, eps – real.  Возвращаемые параметр: currentPoint, currentDistances | Процедура |

# Основные расчетные формулы

Теорема пифагора для расчёта расстояния между двумя точками

Координаты точек: x1, y1 и x2, y2

Теорема Пифагора:

Полученная формула:

Метод бинарного поиска

1. Весь проверяемый отрезок [a, b] разбиваем точкой пополам, её координаты ((a+b)/2, 0).
2. Находим максимальное расстояние от полученной точки до точек слева от неё, максимальное расстояние до точек справа от неё.
3. Проверяем, какое из этих расстояний больше, если слева, то проверяем отрезок [a, (a+b)/2], иначе [(a+b)/2, b]
4. Если длина проверяемого отрезка больше eps, то повторяем с шага 1

# Результаты расчетов и тестирование программы

Тест 1

Тестовая ситуация: Вывод при стандартных исходных данных

Исходные данные:

Eps: 0.0001

Точки: (145, 112)

(33, 168)

(33, 162)

(122, 58)

(78, 98)

(126, 82)

(125, 93)

(114, -58)

(126, -28)

(138, -62)

Ожидаемый результат:

Точка: (33, 0)

Расстояние: 168

Полученный результат:

Рисунок 1 – Тест 1



Рисунок 1 – Тест 1

Тест 2

Тестовая ситуация: Вывод, расстояние зависит от двух точек

Исходные данные:

Eps: 0.0001

Точки: (78, 0)

(78, 0)

(78, 0)

(122, 58)

(78, 98)

(126, 82)

(125, 93)

(114, -58)

(126, -28)

(138, -62)

Ожидаемый результат:

Точка: (91.3404, 0.0000)

Расстояние: 98.9038

Полученный результат:

Рисунок 2 – Тест 2



Рисунок 2 – Тест 2

Тест 3

Тестовая ситуация: Вывод, расстояние относительно 1 точки

Исходные данные:

Eps: 0.0001

Точки: (0, 0)

Ожидаемый результат:

Точка: (0.0000, 0.0000)

Расстояние: 0.0000

Полученный результат:

Рисунок 3 – Тест 3

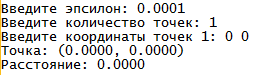


Рисунок 3 – Тест 3

Приложение А

(обязательное)

Исходный код программы

program R5;

{$APPTYPE CONSOLE}

uses

Windows;

type

TPoint = record

x, y: Real;

end;

TPoints = array of TPoint;

TDistances = record

left, right: Real;

end;

var

points: TPoints;

pointLeft, pointRight, currentPoint: TPoint;

eps: Real;

currentDistances: TDistances;

{ Ввод eps }

function getEps():Real;

begin

write('Введите эпсилон: ');

readln(result);

end;

{ Ввод точек с клавиатуры }

function getPoints():TPoints; overload;

var

i, n: Integer;

begin

write('Введите количество точек: ');

readln(n);

setLength(result, n);

for i := 0 to n - 1 do

begin

write('Введите координаты точек ', i + 1, ': ');

readln(result[i].x, result[i].y);

end;

end;

{ Ввод стандартного набора точек }

function getPoints(const str: String):TPoints; overload;

begin

setLength(result, 10);

result[0].x := 145; result[0].y := 112;

result[1].x := 33; result[1].y := 168;

result[2].x := 33; result[2].y := 162;

result[3].x := 122; result[3].y := 58;

result[4].x := 78; result[4].y := 98;

result[5].x := 126; result[5].y := 82;

result[6].x := 125; result[6].y := 93;

result[7].x := 114; result[7].y := -58;

result[8].x := 126; result[8].y := -28;

result[9].x := 138; result[9].y := -62;

end;

{ Расчёт расстояние между точками }

function getDistance(const point1, point2: TPoint):Real;

begin

result := sqrt(sqr(point2.x - point1.x) +

sqr(point2.y - point1.y));

end;

{ Поиск границ проверяемого отрезка }

procedure getSegment(const points: TPoints;

var min, max: TPoint);

var

i: Integer;

begin

min.y := 0;

max.y := 0;

min.x := points[0].x;

max.x := points[0].x;

for i := 1 to length(points) - 1 do

begin

if points[i].x > max.x then

max.x := points[i].x;

if points[i].x < min.x then

min.x := points[i].x;

end;

end;

{ Расчёт максимальных расстояний от точки до точек

слева и справа }

function getMaxOfDistances(const points: TPoints;

const point: TPoint):TDistances;

var

i: Integer;

distance: Real;

begin

result.left := 0;

result.right := 0;

for i := 0 to length(points) - 1 do

begin

distance := getDistance(point, points[i]);

if points[i].x > point.x then

begin

if distance > result.right then

result.right := distance;

end

else

begin

if distance > result.left then

result.left := distance;

end;

end;

end;

{ Расчёт количества символов после точки }

function getOutputLength(eps: Real):Integer;

begin

result := 0;

while eps < 1 do

begin

inc(result);

eps := eps \* 10;

end;

end;

{ Вывод найденной точки и максимального расстояния }

procedure outputResult(const point: TPoint;

const distance: TDistances;

const outputLength: Integer);

begin

writeln('Точка: (', point.x:0:outputLength, ', ',

point.y:0:outputLength, ')');

if distance.left > distance.right then

writeln('Расстояние: ', distance.left:0:outputLength)

else

writeln('Расстояние: ', distance.right:0:outputLength);

end;

{ Поиск нужной точки, используя метод бинарного поиска }

procedure binarySearch( const points: TPoints;

var currentPoint, pointLeft,

pointRight: TPoint;

var currentDistances: TDistances;

const eps: Real);

begin

currentPoint.y := 0;

currentPoint.x := (pointLeft.x + pointRight.x) / 2;

while (pointRight.x - pointLeft.x > eps) do

begin

currentDistances := getMaxOfDistances

(points, currentPoint);

if currentDistances.left < currentDistances.right then

pointLeft.x := currentPoint.x

else

pointRight.x := currentPoint.x;

currentPoint.x := (pointLeft.x + pointRight.x) / 2;

end;

end;

begin

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

eps := getEps();

points := getPoints('default');

getSegment(points, pointLeft, pointRight);

binarySearch(points, currentPoint, pointLeft,

pointRight, currentDistances, eps);

outputResult(currentPoint, currentDistances,

getOutputLength(eps));

readln;

end.