|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ, ФИЛИАЛ ПЛОВДИВ**  **ФАКУЛТЕТ ПО ЕЛЕКТРОНИКА И АВТОМАТИКА** | | | | | | | | | |
|  |  | | | | | | | | | |
| **Дисциплина** | | ***Синтез и анализ на алгоритми*** | | | | | | | | |
| **Уч. година** | | *2021/2022* | **Семестър** | *IV* | **Kурс** | *II* | | **Група** | | 42б |
| **Студент** | | Благовест Боруков | | | | | **Фак.№** | | 377123 | |
|  | |  | | | | | | | | |

**Курсова Работа №1**

Условие: Дадени за n точки в равнината, зададени са с координатите си. Да се намери двойка точки, между които разстоянието е минимално.

Код:

#include <stdio.h>  
#include <float.h>  
#include <stdlib.h>  
#include <math.h>  
  
// Структура за представяне на точка с координати (x;y)  
struct Point  
{  
 int x;  
 int y;  
};  
typedef struct Point Point;  
  
Point firstPoint;  
Point secondPoint;  
  
//Сортиране масив от точки по X  
int compareX(const void\* a, const void\* b)  
{  
 Point \*p1 = (Point \*)a;  
 Point \*p2 = (Point \*)b;  
  
 return (p1->x - p2->x);  
}  
// Сортиране масив от точки по Y  
int compareY(const void\* a, const void\* b)  
{struct  
 Point \*p1 = (Point \*)a;  
 Point \*p2 = (Point \*)b;  
  
 return (p1->y - p2->y);  
}  
  
// Намиране на дистанцията между две точки  
float Distance(Point p1, Point p2)  
{  
 return sqrt((p1.x - p2.x) \* (p1.x - p2.x) + (p1.y - p2.y) \* (p1.y - p2.y));  
}  
  
// Функция връщата най-малка дистанция между две точки в масив P[] с расмер n.  
float Case(Point P[], int n)  
{  
 float min = FLT\_MAX;  
 for (int i = 0; i < n; i++)  
 {  
 for (int j = i+1; j < n; j++)  
 {  
 if (Distance(P[i], P[j]) < min)  
 {  
 min = Distance(P[i], P[j]);  
 firstPoint = P[i];  
 secondPoint = P[j];  
 }  
 }  
 }  
 return min;  
}  
  
// Намиране на по малката от две float стойности  
float Min(float x, float y)  
{  
 return (x < y)? x : y;  
}  
  
// Намиране разстоянието на най-близките точки в strip[] с определен размер.  
// Всички точки в strip[] се сортират по Y. Те имат горна граница на минимилно  
// разстояние d.  
float Strip(Point strip[], int size, float d)  
{  
 float min = d;  
  
 qsort(strip, size, sizeof(Point), compareY);  
  
 // Проверяваме всяка точка докато разликата между Y координата  
 // е по-малка от d.  
 // This is a proven fact that this loop runs at most 6 times  
 for (int i = 0; i < size; i++)  
 {  
 for (int j = i+1; j < size && (strip[j].y - strip[i].y) < min; j++)  
 {  
  
 if (Distance(strip[i],strip[j]) < min)  
 {  
 min = Distance(strip[i], strip[j]);  
 firstPoint = strip[i];  
 secondPoint = strip[j];  
 }  
 }  
 }  
  
 return min;  
}  
  
// Намираме най малкото разстояние с рекурсия. Масива P съдържа  
// всички точки сортирани по X.  
float Rec(Point P[], int n)  
{  
 // Ако точките са 2 или 3 използваме brute force.  
 if (n <= 3)  
 return Case(P, n);  
  
 // Намираме средата.  
 int mid = n/2;  
 Point midPoint = P[mid];  
  
 // Изчисляваме най-малкото разстояние спрямо средата отляво dl и отдясно dr.  
 float dl = Rec(P, mid);  
  
 float dr = Rec(P + mid, n-mid);  
  
 // Намираме по-малкото от двете разстояния.  
 float d = Min(dl, dr);  
  
 // Масив strip[] съдържащ точките които са по-близо до средата от d.  
 Point strip[n];  
 int j = 0;  
  
 for (int i = 0; i < n; i++)  
 {  
 if (abs(P[i].x - midPoint.x) < d)  
 {  
 strip[j] = P[i];  
 j++;  
 }  
 }  
  
 // Намираме най-близките точки в strip[]. Връщаме минималната стойност на d  
 // и май-малкото разстояние е strip[].  
 return Min(d, Strip(strip, j, d) );  
}  
  
// Основната функция която намира най-малкото разстояние.  
// Използваа основно closestUtil().  
float closest(Point P[], int n)  
{  
 qsort(P, n, sizeof(Point), compareX);  
  
 // Използваме рекурсивната функция closestUtil() за да намерим на-малкото разстояние.  
 return Rec(P, n);  
}  
  
int main()  
{  
 Point P[] = {{2, 3}, {2, 4}, {35, 70}, {1, 5}, {15, 10}, {1, 2}, {2, 2}, 35, 69, {100, 100}, {20, 20}};  
 int n = sizeof(P) / sizeof(P[0]);  
 float smallestDist = closest(P, n);  
 printf("The smallest distance is %f between point with coordinates (%d ; %d) and point with coordinates (%d ; %d).",smallestDist , firstPoint.x, firstPoint.y, secondPoint.x,secondPoint.y);  
 return 0;  
}