```
/*
 * main.cpp
 *
 *  Created on: 10/12/2013
 *      Author: isabel
 */


//============================================================================
// Name        : puntoscercanos.cpp
// Author      :
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================

#include <iostream>
#include <cmath>
using namespace std;

struct Punto { double x;
    double y;
};

double absolute(double x) {
      if (x>=0) return x;
    else return -x;
}

double distancia(Punto p1,Punto p2) {
      return (sqrt(double(p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y)));
}

double minimo(double x, double y) {
      double z;
      if (x<=y) z = x; else z=y;
    return z;
}

void solucionDirecta(Punto p[], int c, int f, int indY[], double& d, int& p1, int& p2) {
      double d1,d2,d3;
      if (f==c+1){
            d = distancia(p[c],p[f]);
            if ((p[c].y) <= (p[f].y))
        {indY[c]=c;indY[f]=f;p1=c;p2=f;}
            else
        {indY[c]=f; indY[f]=c; p1=f;p2=c;};
      }
       else if (f==c+2) {
        //Menor distancia y puntos que la producen
        d1=distancia(p[c],p[c+1]); d2=distancia(p[c],p[c+2]);
        d3=distancia(p[c+1],p[c+2]);
        d = minimo(minimo(d1,d2),d3);
        if (d==d1) {p1=c;p2=c+1;}
        else if (d==d2) {p1=c;p2=c+2;}
        else {p1=c+1;p2=c+2;};
        //Ordenar
        if (p[c].y<=p[c+1].y){
            if (p[c+1].y<=p[c+2].y)
            {indY[c]=c;indY[c+1]=c+1;indY[c+2]=c+2;}
            else if (p[c].y<=p[c+2].y)
            {indY[c]=c;indY[c+1]=c+2;indY[c+2]=c+1;}
            else
```

```
            {indY[c]=c+2;indY[c+1]=c;indY[c+2]=c+1;}
        }
        else {
            if (p[c+1].y>p[c+2].y)
            {indY[c]=c+2;indY[c+1]=c+1;indY[c+2]=c;}
            else if (p[c].y>p[c+2].y)
            {indY[c]=c+1;indY[c+1]=c+2;indY[c+2]=c;}
            else
            {indY[c]=c+1;indY[c+1]=c;indY[c+2]=c+2;}
        }}
};

void mezclaOrdenada(Punto p[], int a, int b, int I[]) {
      int u[b-a+1];
      int i,j,k;
      int m = (a+b)/2;
      i=a; j=m+1;k=0;
      while ( (i<=m)&&(j<=b) ) {
            if (p[I[i]].y <= p[I[j]].y) { u[k] = I[i]; i = i+1;}
            else { u[k]=I[j]; j=j+1;}
            k=k+1;
      }
      while (i<=m) {
            u[k] = I[i]; i = i+1; k = k+1;
      }
      while ( j <= b ) {
            u[k] = I[j]; j = j+1; k = k+1;
      }
      for ( k = a ; k <= b ; k++ ) I [ k ] = u [ k -a ] ;
}

void parMasCercano(Punto p[], int c, int f, int indY[], double& d, int& p1, int& p2){
      int m;int i,j,p11,p12,p21,p22;double d1,d2;
      if (f-c+1<4) solucionDirecta(p,c,f,indY,d,p1,p2);
      else {
            m = (c+f)/2;
        parMasCercano(p,c,m,indY,d1,p11,p12);
        parMasCercano(p,m+1,f,indY,d2,p21,p22);
        if (d1<=d2) {d=d1;p1=p11;p2=p12;}
        else {d=d2;p1=p21;p2=p22;};
        //Mezcla ordenada por la y
        mezclaOrdenada(p,c,f,indY);
        //Filtrar la lista
        int aux[f-c+1]; int k = 0;
        for (int i = c; i<= f; i++)
        {
            if (absolute(p[m].x-p[indY[i]].x)<=d) {aux[k]=indY[i];k++;}
        };
        //Calcular las distancias
        i=0;
        // El vector aux tiene informacion hasta la posicion k
        while ( i != f-c+1 && i!=k){
            int count = 1; j = i+1;
            while ((j < f-c+1) && j != k && (count<=7)) {
                double daux = distancia(p[aux[i]],p[aux[j]]);
                if (daux<d) {d=daux; p1=aux[i]; p2=aux[j];}
                count++; j++;
            };
            i++;
        };
    }
};
```

```
int main() {

	cout << "Numero de puntos \n";
	int npuntos;
	cin >> npuntos;
	Punto v[100];
	Punto p;
	for (int i = 0; i<npuntos;i++)
	{
		cin >> p.x >> p.y;
		v[i] = p;
	}
	int I[100]; double d; int p1; int p2;
	parMasCercano(v, 0, npuntos-1, I, d, p1, p2);
	cout << "La distancia minima es: " << d << endl;
	cout << "Entre los puntos " << p1 << "   " << p2;
	return 0;
}
```