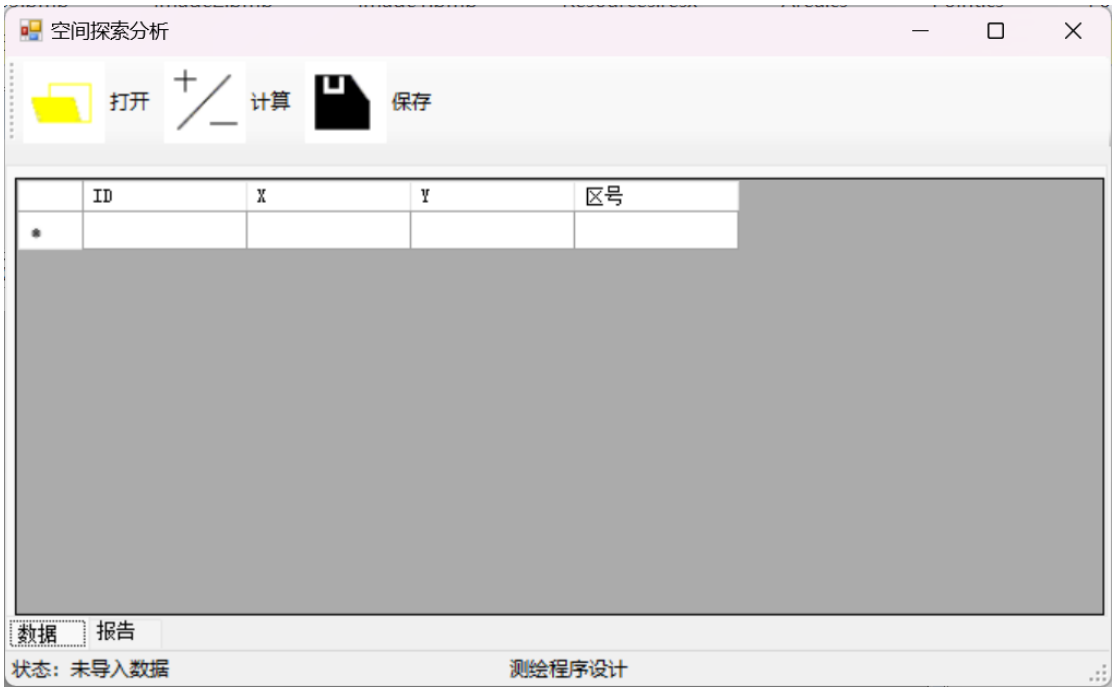
一、程序优化性说明

1．用户交互界面说明（建议 200 字以内，给出主要用户交互界面图）

（1）包括菜单、工具栏、表格等功能。

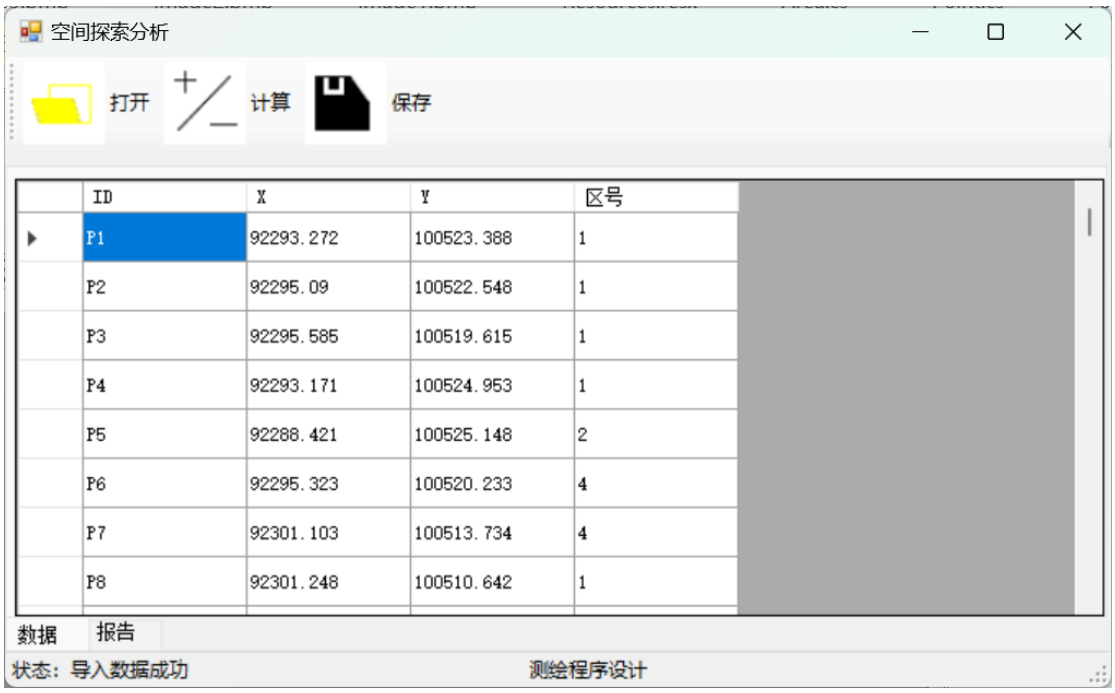
（2）将相关统计信息、计算报告在用户界面中显示；

（3）保存为文本文件（*.txt）。



2．程序运行过程说明（建议 200 字以内，给出程序运行过程截图）

1. 打开数据：


空间探索分析
打开
计算
保存
ID
X
Y
区号
P1 92293.272 100523.388 1
P2 92295.09 100522.548 1
P3 92295.585 100519.615 1
P4 92293.171 100524.953 1
P5 92288.421 100525.148 2
P6 92295.323 100520.233 4
P7 92301.103 100513.734 4
P8 92301.248 100510.642 1
数据
报告
状态：导入数据成功
测绘程序设计

2. 进行运算：


空间探索分析
打开
计算
保存
1，P6的坐标x，92295.323
2，P6的坐标y，100520.233
3，P6的区号，4
4，1区（区号为1）的事件数量n1，1408
5，4区（区号为4）的事件数量n4，288
6，6区（区号为5）的事件数量n6，744
7，事件总数n，7754
8，坐标分量x的平均值X_bar，95635.466
9，坐标分量y的平均值Y_bar，97175.589
10，P6坐标分量与平均中心之间的偏移量a6，-3340.143
11，P6坐标分量与平均中心之间的偏移量b6，3344.644
12，辅助量A，-501728394.42
13，辅助量B，60614732934.584
14，辅助量C，-60612656412.248
15，标准差椭圆长轴与竖直方向的夹角 ，-0.781
16，标准差椭圆的长半轴 ，3954.899
17，标准差椭圆的短半轴 ，3955.035
数据
报告
状态：计算成功
测绘程序设计

3.保存输出：

3．程序运行结果（给出程序运行结果）

二、程序规范性说明

1．程序功能与结构设计说明（建议 500 字以内）

功能：运用空间探索性分析，对数据展开研究。通过标准差椭圆来呈现数据的分布方向与范围，借助空间权重来界定各区间之间的相互关系；利用空间莫兰指 数，判别数据在空间上的分布状况

结构设计说明：
Algo 是主算法的实现；Area 是每个分区的数据结构；Point 是每个犯罪事件点

2. 核心算法源码（给出主要算法的源码）

```
using System;
using System.Collections.Generic;
namespace KongJianTanSuo
{
    class Algo
    {
        #region 根据区号得到对应的分析区的所有点
        public static void GetArea(List<Point> points,double area_code,ref Area area)
        {
            List<Point> pointsCode = new List<Point>();
            foreach(Point d in points)
            {
                if(d.area_code == area_code)
                {
                    pointsCode.Add(d);
                }
            }
            area.points = pointsCode;
        }
        #endregion

        #region 计算XY的平均值
        public static void GetXavgYavg(ref Area area)
        {
            foreach(Point d in area.points)
```

```
            {
                area.Xavg += d.x;
                area.Yavg += d.y;
            }
            area.Xavg /= area.points.Count;
            area.Yavg /= area.points.Count;
        }
        #endregion

        #region 计算椭圆参数
        public static void GetTuoYuan(ref Area area)
        {
            foreach(Point d in area.points)
            {
                d.a = d.x - area.Xavg;
                d.b = d.y - area.Yavg;
            }
            foreach(Point d in area.points)
            {
                area.ai2 += d.a * d.a;
                area.bi2 += d.b * d.b;
                area.aibi += d.a * d.b;
            }
            double A = area.ai2 - area.bi2;
            double B = Math.Sqrt(Math.Pow(area.ai2 - area.bi2, 2) + 4.0 *
Math.Pow(area.aibi, 2));
            double C = 2.0 * area.aibi;
            area.A = A;
            area.B = B;
            area.C = C;
            area.Angle = Math.Atan((A + B) / C);
            double SDEXabove = 0;
            double SDEYabove = 0;
            foreach(Point d in area.points)
            {
                SDEXabove += Math.Pow((d.a * Math.Cos(area.Angle) + d.b *
Math.Sin(area.Angle)),2);
                SDEYabove += Math.Pow((d.a * Math.Sin(area.Angle) + d.b *
Math.Cos(area.Angle)),2);
            }
            area.SDEX = Math.Sqrt(2.0 * (SDEXabove / area.points.Count));
            area.SDEY = Math.Sqrt(2.0 * (SDEYabove / area.points.Count));
        }
        #endregion
```

```
        #region 构建权重矩阵
        public static double[,] GetMatrix(Area area1, Area area2, Area area3, Area
area4, Area area5, Area area6, Area area7)
        {
            GetXavgYavg(ref area1);
            GetXavgYavg(ref area2);
            GetXavgYavg(ref area3);
            GetXavgYavg(ref area4);
            GetXavgYavg(ref area5);
            GetXavgYavg(ref area6);
            GetXavgYavg(ref area7);
            Area[] areas = { area1, area2, area3, area4, area5, area6, area7 };
            double[,] Matrix = new double[7, 7];
            for(int i = 0;i < 7;i++)
            {
                for(int j = 0;j < 7;j++)
                {
                    if(i == j)
                    {
                        Matrix[i, j] = 0;
                        continue;
                    }
                    Matrix[i, j] = 1000.0 / (GetDistance(areas[i], areas[j]));
                }
            }
            return Matrix;
        }
        public static double GetDistance(Area area1,Area area2)
        {
            double temp = Math.Pow(area1.Xavg - area2.Xavg, 2) + Math.Pow(area1.Yavg -
area2.Yavg, 2);
            return Math.Sqrt(temp);

        }
        #endregion

        #region 计算全局莫兰指数
        public static void GetMolan(ref Area areaTotal,Area area1, Area area2, Area
area3, Area area4, Area area5, Area area6, Area area7,ref double[,]Martrix)
        {
            //全局莫兰
            Area[] areas = { area1, area2, area3, area4, area5, area6, area7 };
            double Xavg = areaTotal.points.Count / 7.0;
```

```
            double S0 = 0;
            double Iabove = 0;
            double Iunder = 0;
            for(int i = 0;i < 7;i++)
            {
                Iunder += (areas[i].points.Count - Xavg) * (areas[i].points.Count -
Xavg);
                for(int j = 0;j < 7;j++)
                {
                    S0 += Martrix[i, j];
                    Iabove += Martrix[i, j] * (areas[i].points.Count - Xavg) *
(areas[j].points.Count - Xavg);
                }
            }
            double Molan = 7.0 / S0 * (Iabove / Iunder);
            //局部莫兰
            double[] JuBuMolan = new double[7];
            for (int i = 0; i < 7; i++)
            {
                double Si2 = 0;
                double right = 0;
                for (int j = 0; j < 7;j++)
                {
                    if(j != i)
                    {
                        Si2 += Math.Pow(areas[j].points.Count - Xavg, 2);
                        right += Martrix[i, j] * (areas[j].points.Count - Xavg);
                    }
                }
                Si2 /= (areas[i].points.Count - 1.0);
                JuBuMolan[i] = (areas[i].points.Count - Xavg) / Si2 * right;
            }
            areaTotal.JuBuMolan = JuBuMolan;
        }
        #endregion

        #region 计算Z得分
        public static void GetZ(ref Area areaTotal)
        {
            double u = 0;
            foreach(double d in areaTotal.JuBuMolan)
            {
                u += d;
            }
```

```
            u /= 7.0;
            double fai = 0;
            foreach(double d in areaTotal.JuBuMolan)
            {
                fai += Math.Pow(d - u, 2);
            }
            fai = Math.Sqrt(fai / 6.0);
            double[] Z = new double[7];
            for(int i = 0;i < 7;i++)
            {
                Z[i] = (areaTotal.JuBuMolan[i] - u) / fai;
            }
        }
        #endregion
    }
}
```