报告文档

1. 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）

程序采用C# WinForm框架进行编写，界面采用Windows标准窗体风格，整个界面从左到右、从上到下依次为：标题栏、菜单栏、工具栏、数据区域、报告区域、状态栏。界面设计简介、符合人性化设计。具体界面如图 1所示。

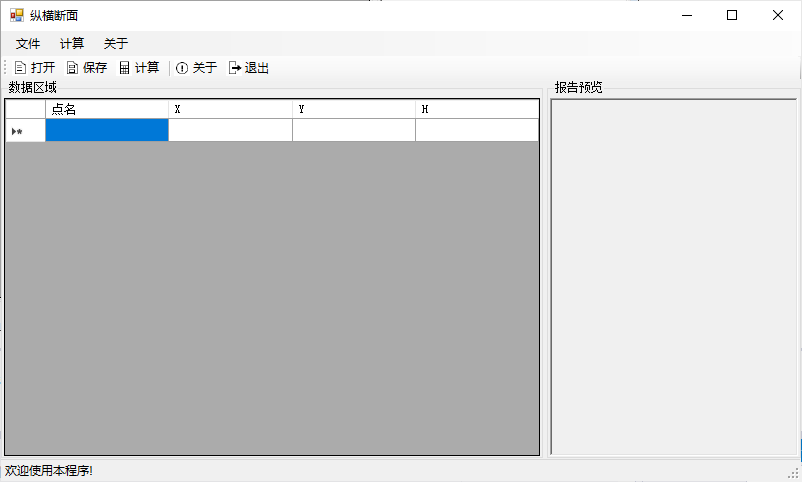


图 1 程序界面

* 1. 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. 点击【打开】，用户选择数据文件，点击【确定】后，读取数据文件内容，并展示到界面

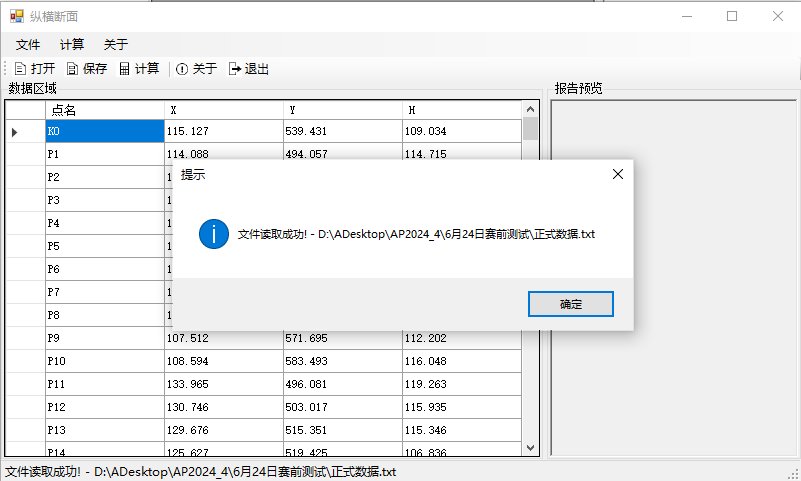


图 2 读取文件

1. 点击【计算】，程序根据读取的数据计算纵横断面相关参数，并生成报告预览。

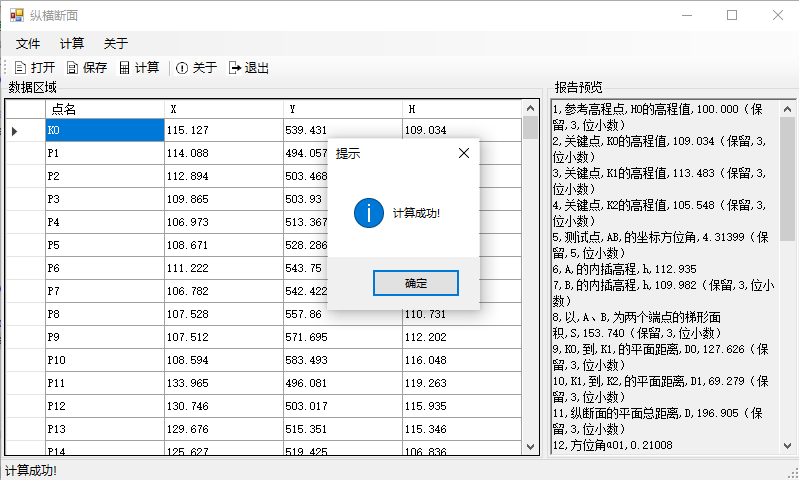


图 3 计算成功

1. 点击【保存】，用户选择保存文件路径，点击【确定】后，保存报告内容到文件。

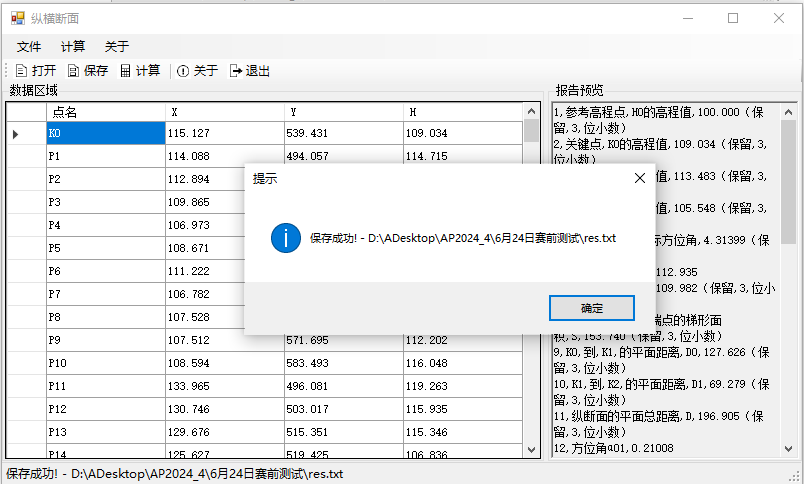


图 4 保存报告

1. 点击【退出】，用户经过二次确认后，退出程序。

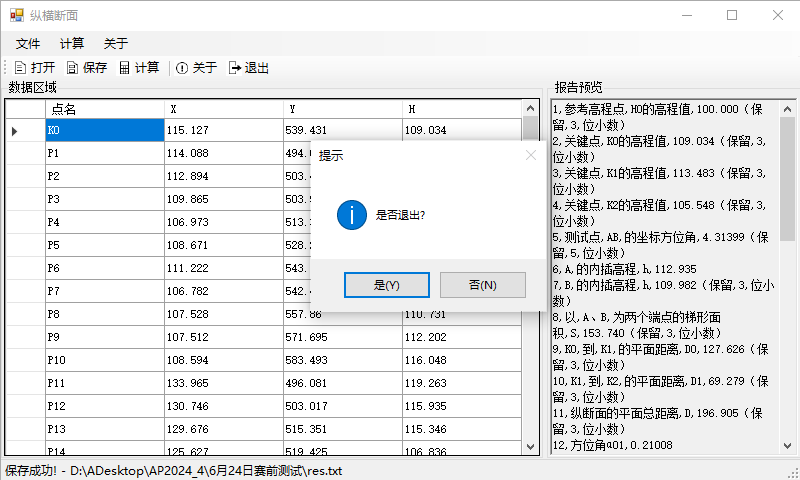


图 5 是否退出

* 1. 程序运行结果（给出程序运行结果）

程序运行结果：

1,参考高程点,H0的高程值,100.000（保留,3,位小数）

2,关键点,K0的高程值,109.034（保留,3,位小数）

3,关键点,K1的高程值,113.483（保留,3,位小数）

4,关键点,K2的高程值,105.548（保留,3,位小数）

5,测试点,AB,的坐标方位角,4.31399（保留,5,位小数）

6,A,的内插高程,h,112.935

7,B,的内插高程,h,109.982（保留,3,位小数）

8,以,A、B,为两个端点的梯形面积,S,153.740（保留,3,位小数）

9,K0,到,K1,的平面距离,D0,127.626（保留,3,位小数）

10,K1,到,K2,的平面距离,D1,69.279（保留,3,位小数）

11,纵断面的平面总距离,D,196.905（保留,3,位小数）

12,方位角ɑ01,0.21008

13,方位角ɑ12,0.41739（保留,5,位小数）

14,第一条纵断面的内插点Z3的坐标X,144.467（保留,3,位小数）

15,第一条纵断面的内插点Z3的坐标Y,545.687（保留,3,位小数）

16,第一条纵断面的内插点Z3的高程H,115.825（保留,3,位小数）

17,第二条纵断面的内插点Y3的坐标X,260.400（保留,3,位小数）

18,第二条纵断面的内插点Y3的坐标Y,575.116（保留,3,位小数）

19,第二条纵断面的内插点Y3的高程H,114.385（保留,3,位小数）

20,第一条纵断面面积S1,1915.468（保留,3,位小数）

21,第二条纵断面面积S2,795.857（保留,3,位小数）

22,纵断面总面积S,2711.325（保留,3,位小数）

23,第一条横断面内插点Q3的坐标X,178.163

24,第一条横断面内插点Q3的坐标Y,549.804（保留,3,位小数）

25,第一条横断面内插点Q3的高程H,114.163（保留,3,位小数）

26,第二条横断面内插点W3的坐标X,272.829（保留,3,位小数）

27,第二条横断面内插点W3的坐标Y,577.346（保留,3,位小数）

28,第二条横断面内插点W3的高程H,110.473（保留,3,位小数）

29,第一条横断面的面积Srow1,138.637（保留,3,位小数）

30,第一条横断面的面积Srow2,111.861（保留,3,位小数）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）

程序可以实现读取数据文件，一键计算纵断面长度、面积、横断面长度、面积等功能。 具体共按菜单栏介绍见表 1。

表 1 程序功能

|  |  |
| --- | --- |
| 菜单栏选项 | 功能简介 |
| 打开 | 用户选择原始数据文件后，读取数据并展示到界面 |
| 保存 | 用户选择保存文件路径后，保存报告到文件 |
| 打开数据文件夹 | 如果用户打开过原始数据，则打开原始数据所在的文件夹 |
| 打开报告文件夹 | 如果用户保存过报告文件，则打开报告文件所在的文件夹 |
| 计算 | 一键计算纵横断面，并生成报告预览 |
| 帮助 | 打开帮助文档 |
| 关于 | 显示程序信息 |
| 退出 | 经过二次确认后退出程序 |

程序共设计有MyData、MyFile、MyH、MyV、MyPoint五个类，程序结构设计清晰，具体类的功能见表 2。

表 2 类的功能

|  |  |
| --- | --- |
| 类名 | 功能简介 |
| MyData | 存放维持窗体运行的必要数据，如inFile、outFile |
| MyFile | 读取数据文件和保存数据文件 |
| MyH | 存放横断面相关信息，如横断面的角度、横断面上的点、横断面的中心点、同时有计算横断长度、面积的相关函数 |
| MyV | 存放纵断面相关信息，如纵断面上的插值点，纵断面长度、纵断面面积、纵断面上的横断面 |
| MyPoint | 点类，存放点的相关信息，同时有计算坐标方位角、计算距离、计算面积的函数。 |

* 1. 核心算法源码（给出主要算法的源码）

1. 计算与另一个点之间的坐标方位角

/// <summary>

/// 计算与另一个点之间的坐标方位角

/// </summary>

/// <param name="OP"></param>

public double AZToOther(MyPoint OP)

{

double dx = OP.X - this.X;

double dy = OP.Y - this.Y;

return Math.Atan2(dy, dx) + (dy < 0 ? 1 : 0) \* 2 \* Math.PI;

}

1. 计算两点之间的距离

/// <summary>

/// 计算与另一个点之间的距离

/// </summary>

/// <param name="OP"></param>

/// <returns></returns>

public double DistToOther(MyPoint OP)

{

double dx = OP.X - this.X;

double dy = OP.Y - this.Y;

return Math.Sqrt(dx \* dx + dy \* dy);

}

1. 高程插值

/// <summary>

/// 根据PList,插值当前点高程

/// </summary>

/// <param name="PList"></param>

/// <returns></returns>

public double InterPH(List<MyPoint> PList)

{

List<MyPoint> temp = new List<MyPoint>();

foreach (var item in PList)

{

if (item.Name == this.Name) continue;

item.Dist = item.DistToOther(this);

temp.Add(item);

}

temp = temp.OrderBy(t => t.Dist).ToList();

//选取最近的五个点计算

double up = 0;

double down = 0;

for (int i = 0; i < 5; i++)

{

up += temp[i].H / temp[i].Dist;

down += 1 / temp[i].Dist;

}

this.H = up / down;

return this.H;

}

1. 计算与另一个点形成的面积

/// <summary>

/// 计算与另一个点形成的面积

/// </summary>

/// <returns></returns>

public double CalS(MyPoint OP,double H0)

{

return (this.H + OP.H - 2 \* H0) / 2 \* (this.DistToOther(OP));

}

1. 计算纵断面长度

/// <summary>

/// 计算纵断面长度

/// </summary>

/// <returns></returns>

public void CalD()

{

this.D = 0;

DList = new List<double>();

for (int i = 0; i < Kp.Count - 1; i++)

{

double temp = Kp[i].DistToOther(Kp[i + 1]);

D += temp;

DList.Add(temp);

}

}

1. 计算纵断面面积

/// <summary>

/// 计算纵断面面积

/// </summary>

/// <param name="H0"></param>

public void CalS(double H0)

{

double temps = 0;

SList = new List<double>();

for (int i = 0; i < NewPList.Count; i++)

{

if (i != NewPList.Count - 1)//最后一个点

{

temps += NewPList[i].CalS(NewPList[i + 1], H0);

}

if (NewPList[i].Name.Contains('K') && i != 0)

{

SList.Add(temps);

S += temps;

//刷新temps

if (i != NewPList.Count - 1)

{

temps = NewPList[i].CalS(NewPList[i + 1], H0);

}

}

}

}

1. 纵断面插入点

/// <summary>

/// 进行内插点

/// </summary>

public void interPoint()

{

double delta = 10;

double L = 0;//剩余的距离

for (int i = 0; i < Kp.Count - 1; i++)

{

double alpha = Kp[i + 1].AZToOther(Kp[i]);

this.alphaList.Add(Kp[i].AZToOther(Kp[i + 1]));

this.NewPList.Add(Kp[i]);

L += Kp[i].DistToOther(Kp[i + 1]);

int count = 0;

while (L > delta)

{

count += 1;

L -= delta;

double x = Kp[i + 1].X + L \* Math.Cos(alpha);

double y = Kp[i + 1].Y + L \* Math.Sin(alpha);

MyPoint mp = new MyPoint();

if (i == 0)

{

mp = new MyPoint($"Z{count}", x, y, 0);

}

if (i == 1)

{

mp = new MyPoint($"Y{count}", x, y, 0);

}

mp.InterPH(this.obsP);

this.NewPList.Add(mp);

}

}

this.NewPList.Add(Kp[Kp.Count - 1]);

}

1. 单个横断面插入点

public void interP(List<MyPoint> obsP, double delta = 5)

{

PList = new List<MyPoint>();

int count = 0;

for (int i = -5; i <= 5; i++)

{

if (i == 0)

{

PList.Add(CP);

continue;

}

count += 1;

double xj = CP.X + i \* Math.Cos(alpha);

double yj = CP.Y + i \* Math.Sin(alpha);

MyPoint mp = new MyPoint($"{count}", xj, yj, 0);

mp.InterPH(obsP);

PList.Add(mp);

}

}

1. 计算单个横断面面积

/// <summary>

/// 计算横断面面积

/// </summary>

/// <param name="H0"></param>

public void CalS(double H0)

{

this.S = 0;

for (int i = 0; i < PList.Count - 1; i++)

{

this.S += PList[i].CalS(PList[i + 1], H0);

}

}