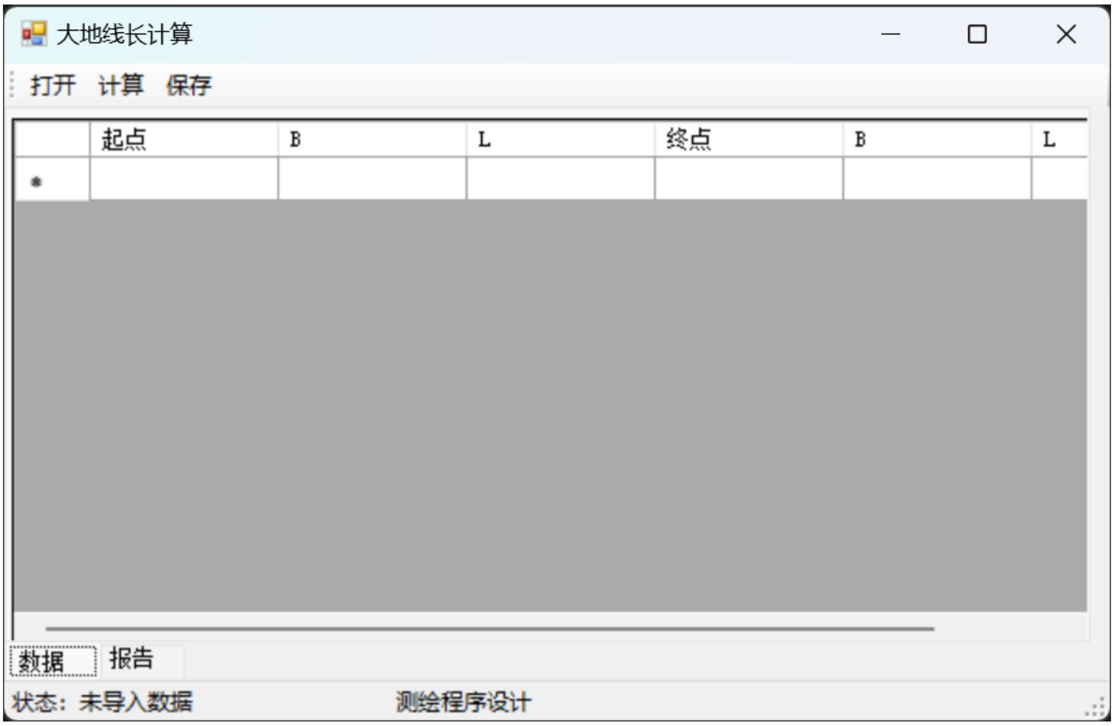
一、程序优化性说明

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）

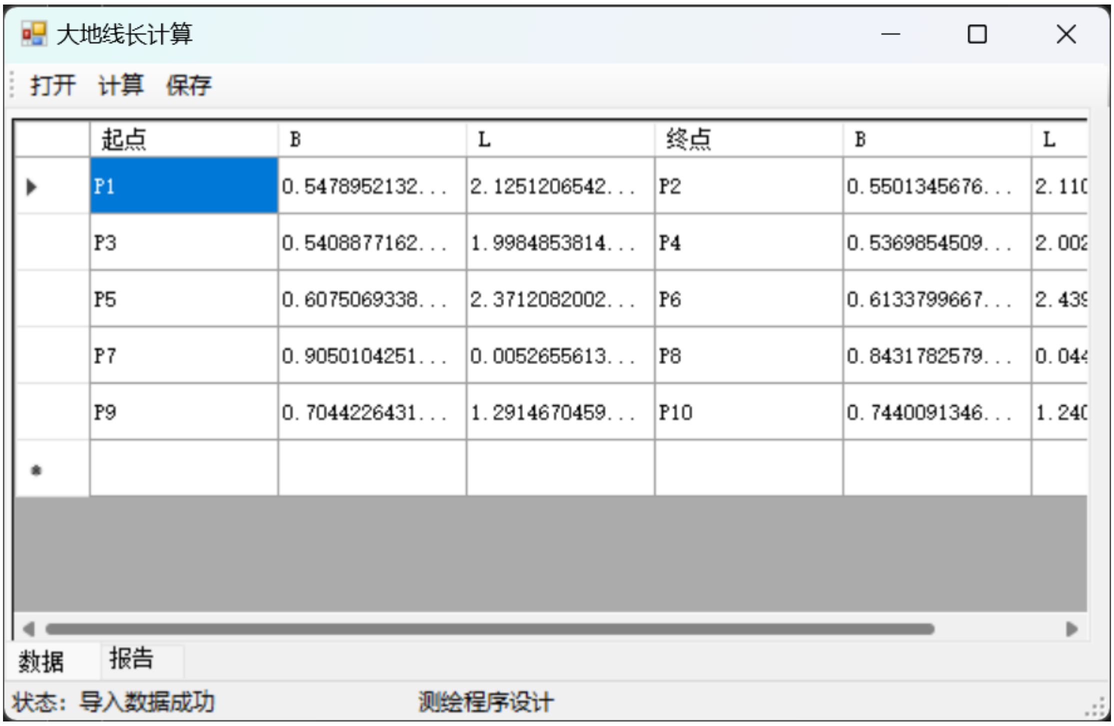


	起点	B	L	终点	B	L
*						

状态：未导入数据 测绘程序设计

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）

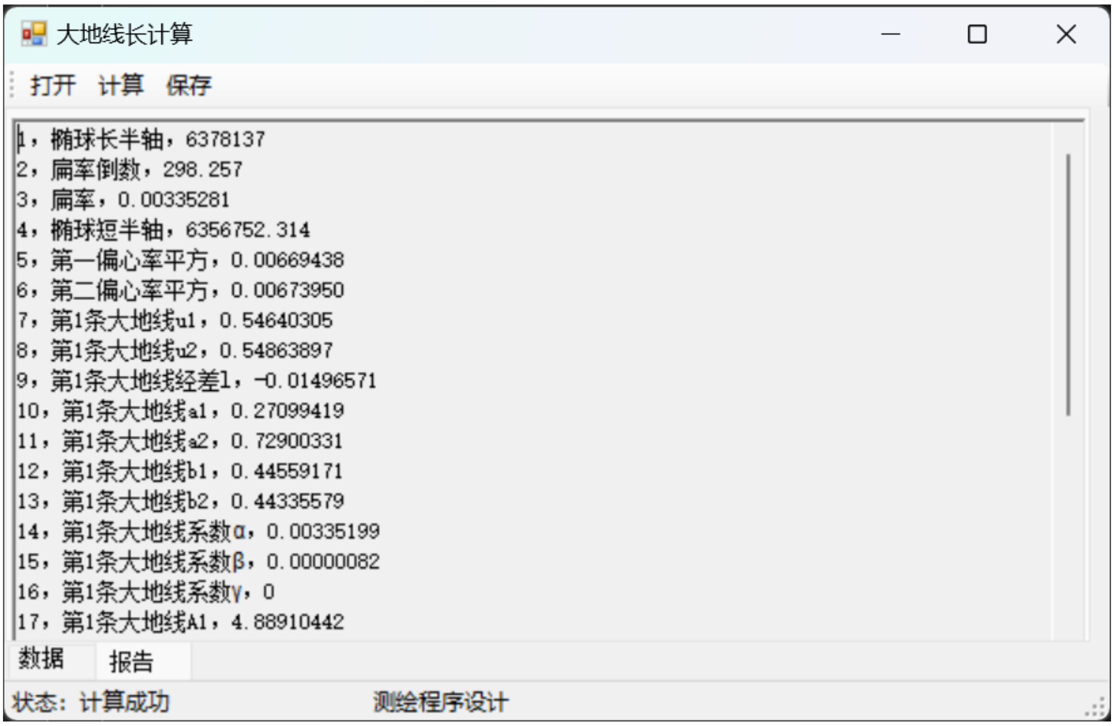
打开：



	起点	B	L	终点	B	L
▶	P1	0. 5478952132...	2. 1251206542...	P2	0. 5501345676...	2. 110
	P3	0. 5408877162...	1. 9984853814...	P4	0. 5369854509...	2. 002
	P5	0. 6075069338...	2. 3712082002...	P6	0. 6133799667...	2. 439
	P7	0. 9050104251...	0. 0052655613...	P8	0. 8431782579...	0. 044
	P9	0. 7044226431...	1. 2914670459...	P10	0. 7440091346...	1. 240
*						

状态：导入数据成功

计算：



```
1，椭球长半轴，6378137
2，扁率倒数，298.257
3，扁率，0.00335281
4，椭球短半轴，6356752.314
5，第一偏心率平方，0.00669438
6，第二偏心率平方，0.00673950
7，第1条大地线u1，0.54640305
8，第1条大地线u2，0.54863897
9，第1条大地线经差l，-0.01496571
10，第1条大地线a1，0.27099419
11，第1条大地线a2，0.72900331
12，第1条大地线b1，0.44559171
13，第1条大地线b2，0.44335579
14，第1条大地线系数α，0.00335199
15，第1条大地线系数β，0.00000082
16，第1条大地线系数γ，0
17，第1条大地线A1，4.88910442
```

状态：计算成功

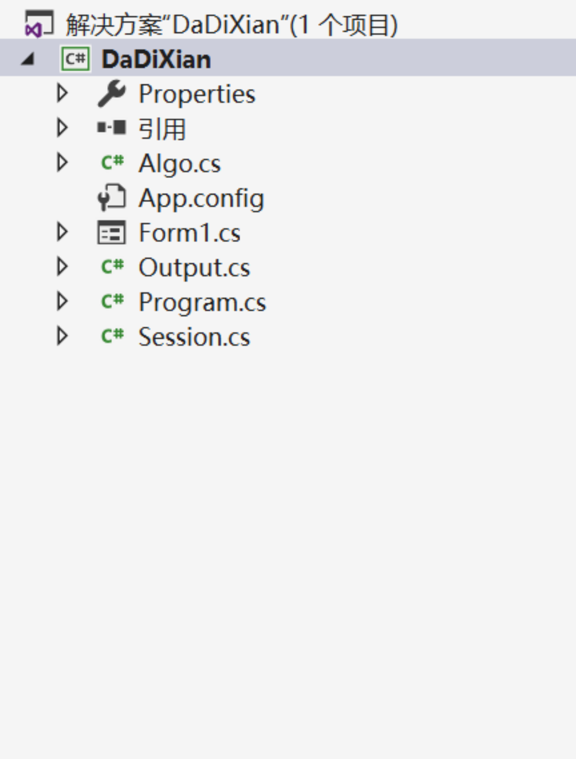
3．程序运行结果（给出程序运行结果）

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：已知大地线起点 P1 的大地坐标（B1，L1）、终点 P2 的大地坐标（B2,L2）。计算得到大地线长度 S。输出中间结果以及最终结果数据并且保存为 txt 文件。

结构设计：Algo 为主要算法，Output 为结果数据输出的类，Session 是每条大地线的数据结构。



```
解决方案"DaDiXian"(1 个项目)
◢ DaDiXian
    ▷ Properties
    ▷ 引用
    ▷ Algo.cs
      App.config
    ▷ Form1.cs
    ▷ Output.cs
    ▷ Program.cs
    ▷ Session.cs
```

2. 核心算法源码（给出主要算法的源码）

```
using System;
using System.Collections.Generic;
namespace DaDiXian
{
    class Algo
    {
        #region 辅助计算
```

```
public static void CalStep12(ref List<Session> sessions,double e2)
{
    foreach(Session d in sessions)
    {
        d.u1 = Math.Atan(Math.Sqrt(1.0 - e2) * Math.Tan(d.B1));
        d.u2 = Math.Atan(Math.Sqrt(1.0 - e2) * Math.Tan(d.B2));
        d.l = d.L2 - d.L1;
        d.a1 = Math.Sin(d.u1) * Math.Sin(d.u2);
        d.a2 = Math.Cos(d.u1) * Math.Cos(d.u2);
        d.b1 = Math.Cos(d.u1) * Math.Sin(d.u2);
        d.b2 = Math.Sin(d.u1) * Math.Cos(d.u2);
    }
}
#endregion

#region 计算起点大地方位角
public static void CalStep2(ref List<Session> sessions,double e2)
{
    foreach(Session d in sessions)
    {
        double start = 0;
        while(true)
        {
            double nameda = d.l + start;
            double p = Math.Cos(d.u2) * Math.Sin(nameda);
            double q = d.b1 - d.b2 * Math.Cos(nameda);
            double A1 = Math.Atan(p / q);
            if(p > 0)
            {
                if(q > 0)
                {
                    A1 = Math.Abs(A1);
                }
                else
                {
                    A1 = Math.PI - Math.Abs(A1);
                }
            }
            else
            {
                if(q > 0)
                {
                    A1 = 2.0 * Math.PI - Math.Abs(A1);
                }
```

```
                        else
                        {
                            A1 = Math.PI + Math.Abs(A1);
                        }
                    }
                    if(A1 < 0)
                    {
                        A1 += 2.0 * Math.PI;
                    }
                    if(A1 > 2.0 * Math.PI)
                    {
                        A1 -= 2.0 * Math.PI;
                    }
                    double sinfai = p * Math.Sin(A1) + q * Math.Cos(A1);
                    double cosfai = d.a1 + d.a2 * Math.Cos(nameda);
                    double fai = Math.Atan(sinfai / cosfai);
                    if(cosfai > 0)
                    {
                        fai = Math.Abs(fai);
                    }
                    else
                    {
                        fai = Math.PI - Math.Abs(fai);
                    }
                    double sinA0 = Math.Cos(d.u1) * Math.Sin(A1);
                    double fai1 = Math.Atan(Math.Tan(d.u1) / Math.Cos(A1));
                    double cosA02 = 1.0 - sinA0 * sinA0;
                    double alpha = e2 / 2.0 + e2 * e2 / 8.0 + e2 * e2 * e2 / 16.0;
                    alpha = alpha - (e2 * e2 / 16.0 + e2 * e2 * e2 / 16.0) * cosA02 +
(3.0 * e2 * e2 * e2) / 128.0 * cosA02 * cosA02;
                    double beta = (e2 * e2 / 16.0 + e2 * e2 * e2 / 16.0) * cosA02 - (e2
* e2 * e2) / 32.0 * cosA02 * cosA02;
                    double gama = e2 * e2 * e2 / 256.0 * cosA02 * cosA02; ;
                    double end = (alpha * fai + beta * Math.Cos(2.0 * fai1 + fai) *
Math.Sin(fai) + gama * Math.Sin(2.0 * fai) * Math.Cos(4.0 * fai1 + 2.0 * fai)) * sinA0;
                    if(Math.Abs(end - start) < 1.0 * Math.Pow(10, -10))
                    {
                        d.nameda = nameda;
                        d.A1 = A1;
                        d.fai = fai;
                        d.SinA0 = sinA0;
                        d.alpha = alpha;
                        d.beta = beta;
                        d.gama = gama;
                        d.fai1 = fai1;
                        break;
                    }
                    else
                    {
                        start = end;
                        continue;
                    }
                }
            }
        }
        #endregion

        #region 计算大地线长度
        public static void CalLength(ref List<Session> sessions,double ep2,double b)
        {
            foreach(Session d in sessions)
            {
                double fai1 = Math.Atan(Math.Tan(d.u1) / Math.Cos(d.A1));
                double cosA02 = 1.0 - d.SinA0 * d.SinA0;
                double k2 = ep2 * cosA02;
                double A = 1.0 - k2 / 4.0 + 7.0 * k2 * k2 / 64.0 - 15.0 * k2 * k2 * k2
/ 256.0;
                A = A / b;
                double B = k2 / 4.0 - k2 * k2 / 8.0 + 37.0 * k2 * k2 * k2 / 512.0;
                double C = k2 * k2 / 128.0 - k2 * k2 * k2 / 128.0;
                double Xs = C * Math.Sin(2.0 * d.fai) * Math.Cos(4.0 * fai1 + 2.0 *
d.fai);
                double S = (d.fai - B * Math.Sin(d.fai) * Math.Cos(2.0 * fai1 + d.fai)
- Xs) / A;

                d.A = A;
                d.B = B;
                d.C = C;
                d.fai1 = fai1;
                d.S = S;
            }
        }
        #endregion
    }
}
```