**流程**

**总览：**

**一．工具类的创建（MenuStrip，ToolStrip），**

**二．不可见控件（openFileDialog，saveFileDialog）**

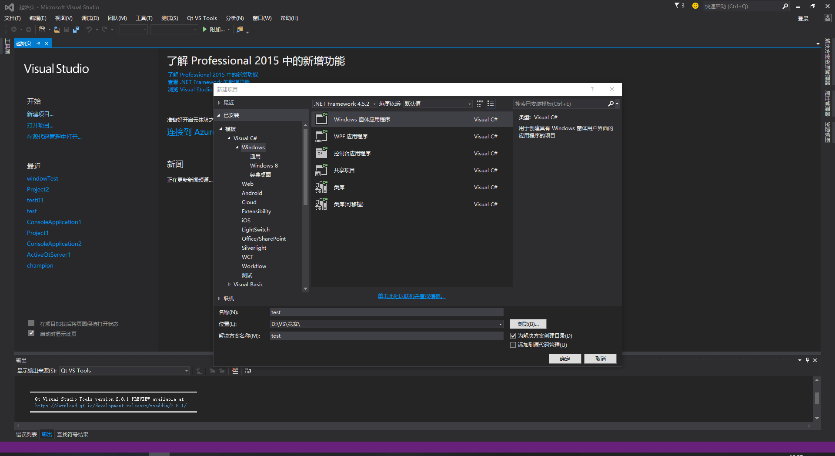
**三．页面设计（TabContral’3个页面’）**

**1. 数据显示页（dataGridView）**

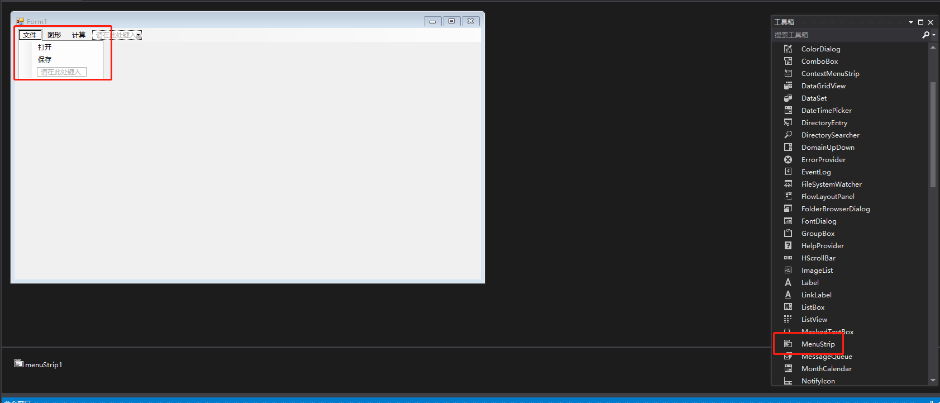
**2. 图形显示页（picturebox）**

**3. 报告显示页（textbox）**

**1.新建Windows窗体。**

****

**2.在工具箱中找到MenuStrip控件，加入Form1中，并编辑菜单栏，加入应有的项目。**

****

**3.在菜单栏中，双击你想要添加功能的按钮，在跳转的界面中进行代码的编辑。**

**如双击文件中的打开，在代码编辑区域加入读取文件的代码，并让其显示在数据框中的控件上，并添加openFileDialog以及saveFileDialog，用于代码部分读取和存储**

**代码1：**

1. **private** **void** 打开ToolStripMenuItem\_Click(object sender, EventArgs e)
2. {
3. dataGridView1.Rows.Clear();
4. pictureBox1.Image = null;
5. **try**
6. {
7. openFileDialog2.Title = "文件打开";
8. openFileDialog2.Filter = "文本文档(\*.txt)|\*.txt";
9. **if** (openFileDialog2.ShowDialog() == DialogResult.OK)
10. {
11. string filepath = openFileDialog2.FileName;
12. string line;
13. string[] strs;//创建一个数组
15. StreamReader sr = **new** StreamReader(filepath);//将文件数据导入
16. line = sr.ReadLine();//将获取的数据放入string类型line
17. strs = line.Split(',');//以,分割获取的数据
18. a = **double**.Parse(strs[1]);//将分割的第二个数据放入a
20. line = sr.ReadLine();
21. strs = line.Split(',');
22. f = **double**.Parse(strs[1]);
24. line = sr.ReadLine();
25. strs = line.Split(',');
26. L0 = **double**.Parse(strs[1]);
28. line = sr.ReadLine();//略过空行

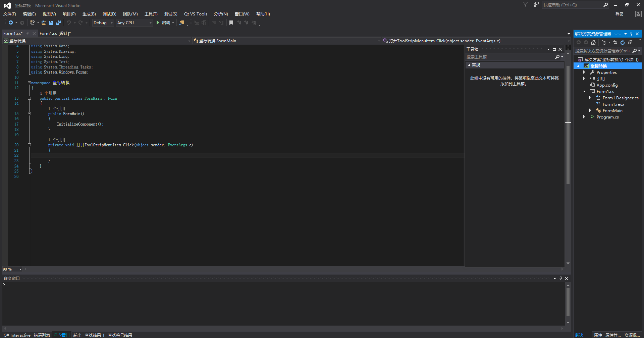
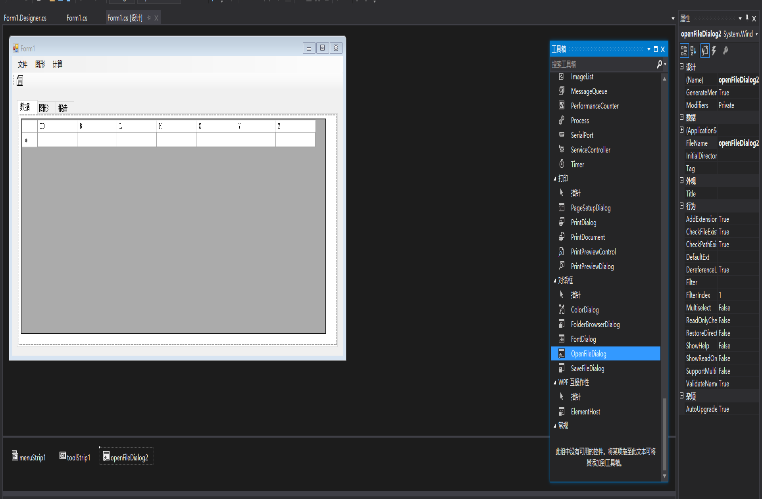
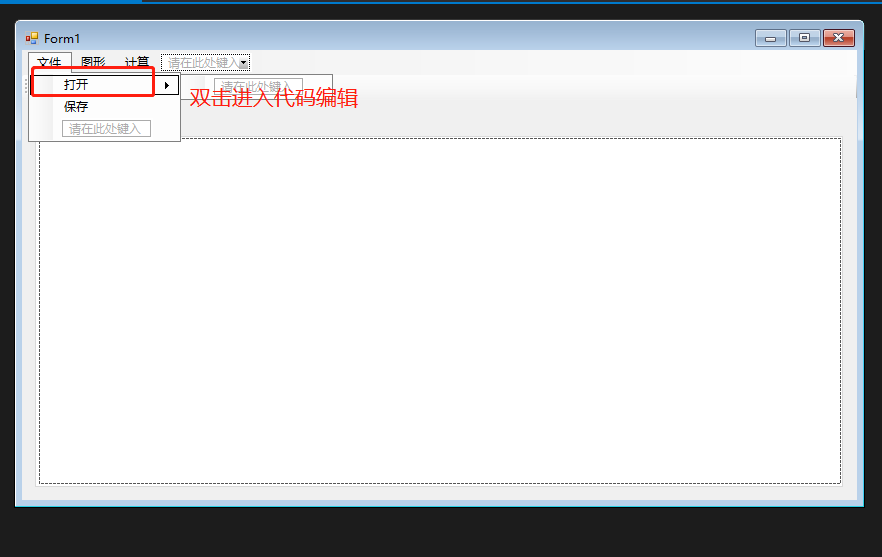
31. s = 0;
32. **while** ((line = sr.ReadLine()) != null)//将ID，B，L，H放入库中
33. {
34. //line = sr.ReadLine();
35. strs = line.Split(',');
36. //dataGrid显示
37. dataGridView1.Rows.Add();
38. dataGridView1.Rows[s].Cells[0].Value = strs[0];
39. dataGridView1.Rows[s].Cells[1].Value = strs[1];
40. dataGridView1.Rows[s].Cells[2].Value = strs[2];
41. dataGridView1.Rows[s].Cells[3].Value = strs[3];
43. point\_ID.Add(strs[0]);//ID
44. point\_B.Add(**double**.Parse(strs[1]));//B
45. point\_L.Add(**double**.Parse(strs[2]));//L
46. point\_H.Add(**double**.Parse(strs[3]));//H
47. s++;
49. }
51. b = a \* (1 - 1 / f);//计算短半轴


55. e1 = ((a \* a) - (b \* b)) / (a \* a);//e1
56. e2 = (e1 \* e1) / (1 - (e1 \* e1));//e2
58. M0 = a \* (1 - (e1 \* e1));//子午圈赤道处的曲率半径

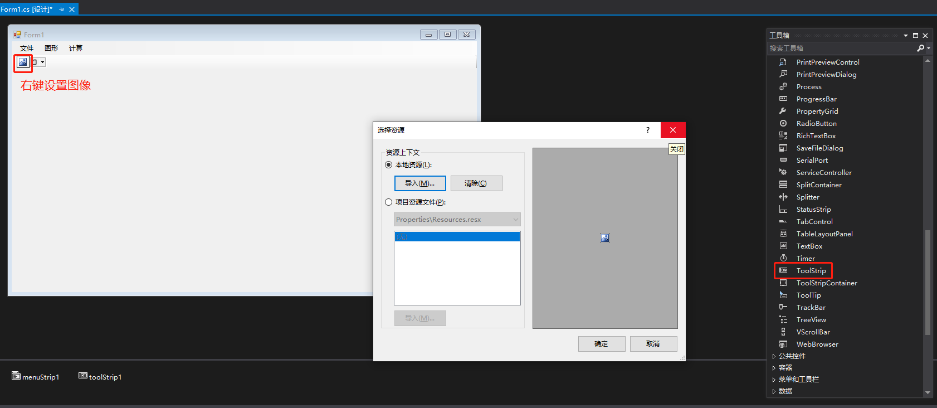




65. }
67. }
68. **catch**(Exception ex)
69. {
70. MessageBox.Show(ex.Message);
71. **return**;
72. }
74. }

****

**4.在工具栏中找到ToolStrip控件，加入Form1中，并编辑工具栏，加入应有的项目。**

****

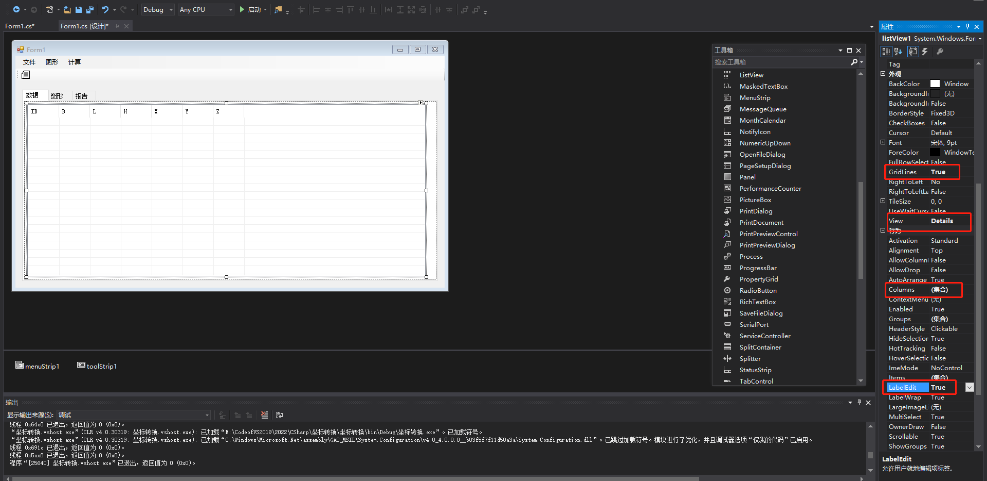
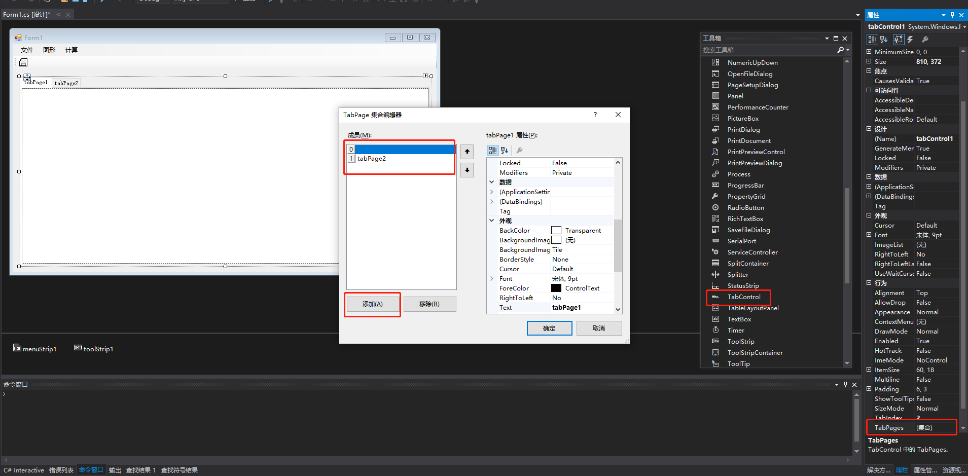
**直接调用MenuStrip的打开函数**

**代码2：**

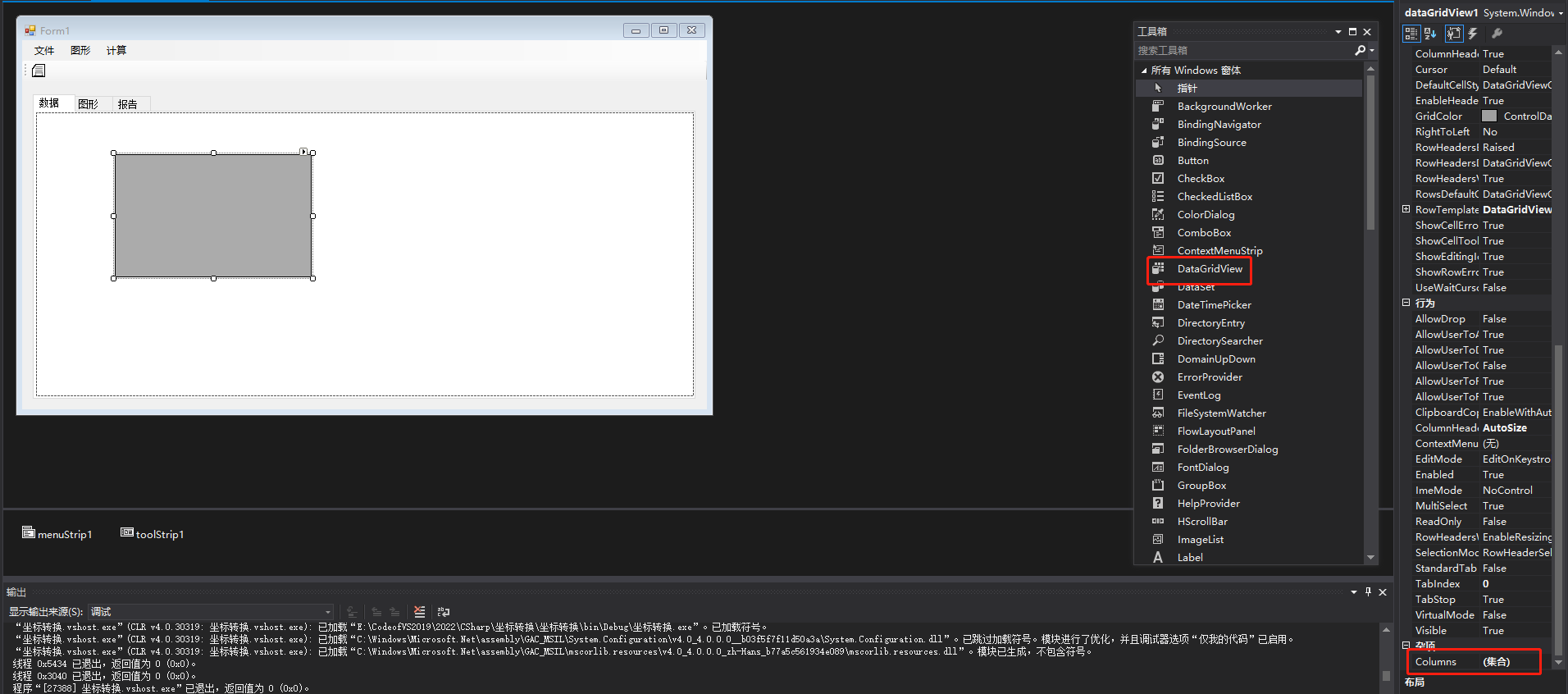
1. **private** **void** toolStripButton1\_Click(object sender, EventArgs e)
2. {
3. 打开ToolStripMenuItem\_Click(sender,e);
4. }

**5.在工具箱中找到tabcontrol，拖入form1中。**

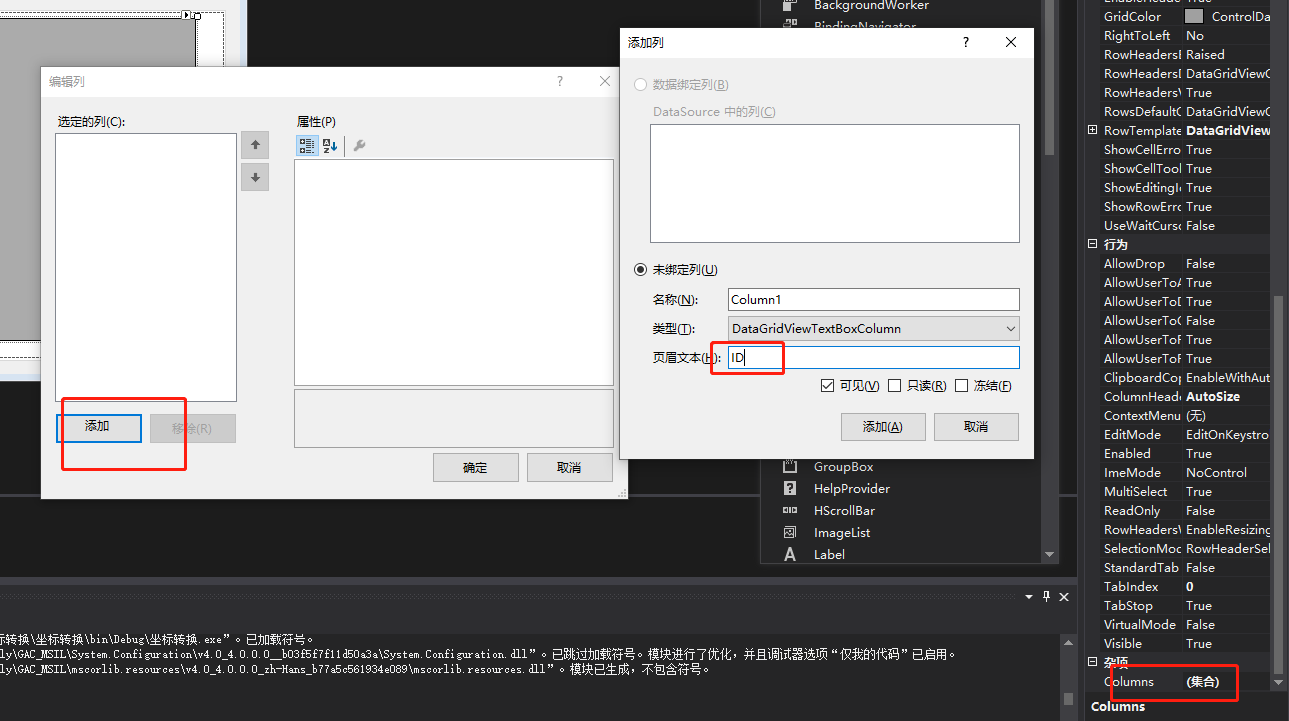
**6.在tabcontrol属性面板中找到TabPages，打开TabPage集合编辑器，在编辑器中编辑每个页面的名称，信息等。数据，图形，报告都是必须项。数据用来显示你的已知数据，计算完成后的计算结果等。图形是用来根据你的计算结果得到的图形报告。报告是用来总结你的计算结果，并且报告要可以输出txt文件。**

****

**7.tabcontrol中第一个界面（数据）中加入dataGridView控件。**

****

**8.在dataGridView的属性面板中，找到Columns，打开编辑列，可根据自己需求更改信息。（添加点名，X，Y，Z等用来显示数据的东西）**

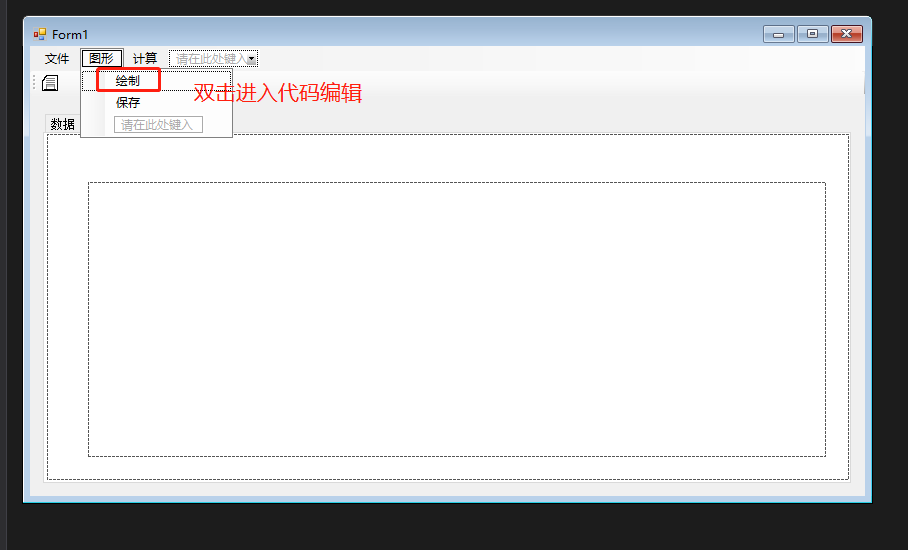
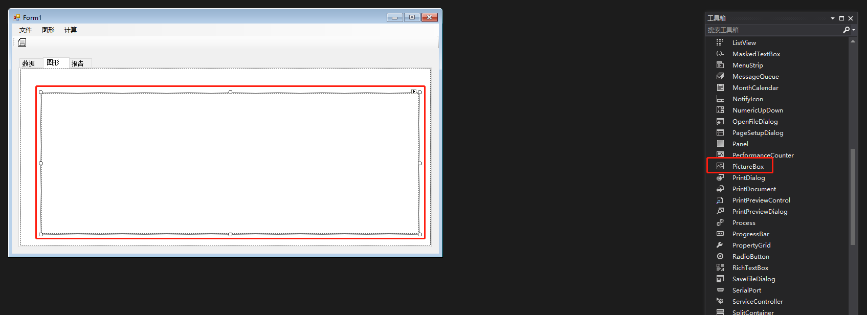
****

**显示代码集成于文件读取中**

**代码：**

**见代码1-37行**

**9.tabcontrol中第二个界面（图形）加入pictureBox。关于图形，大致步骤为：创建画板（pictureBox），创建画笔，定义起点和终点进行画图。给出下列图片与代码进行提示。**

****

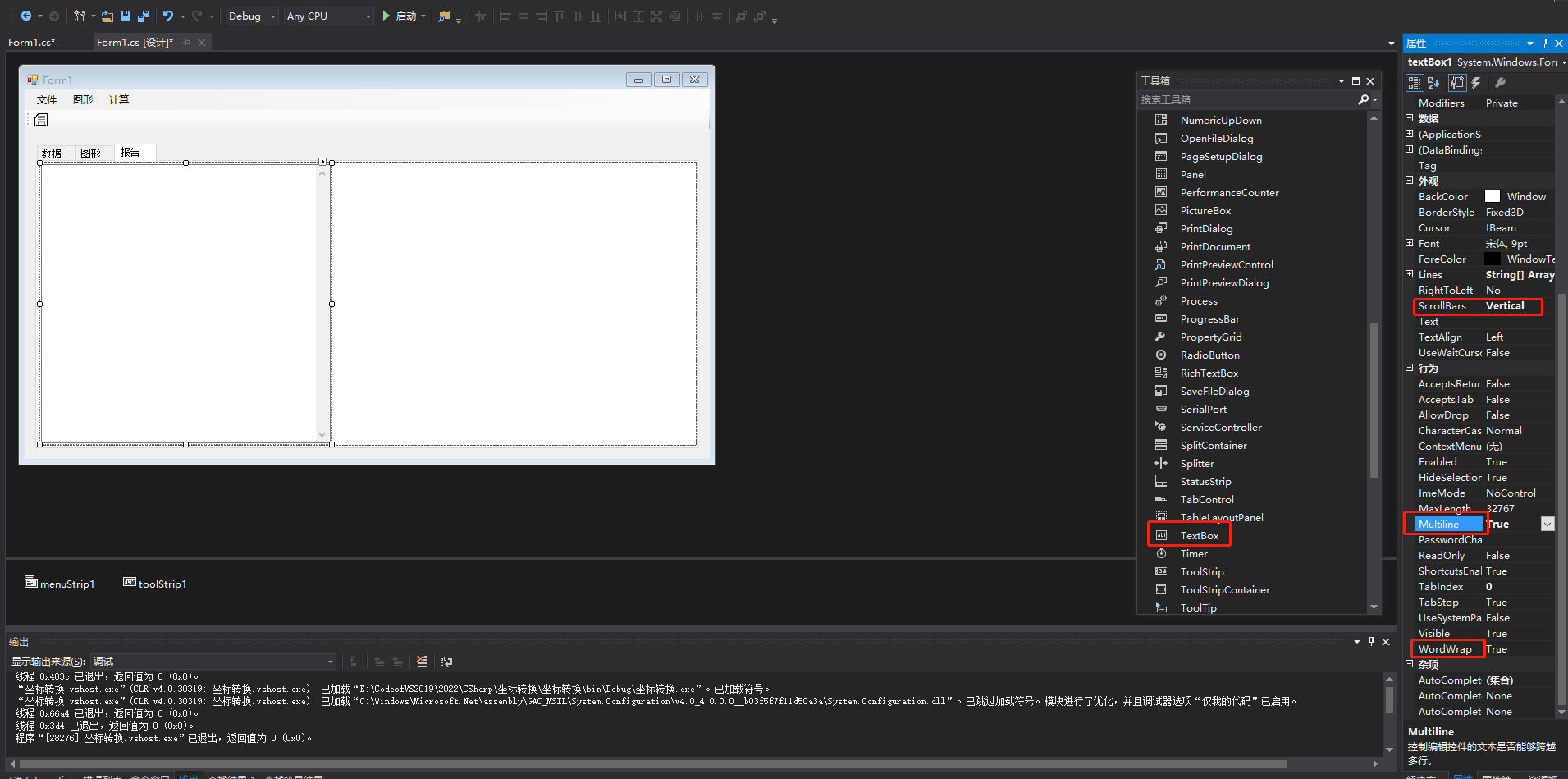
**此题目未要求画图，做个简单示例**

**代码3：**

1. **private** **void** 绘制ToolStripMenuItem\_Click(object sender, EventArgs e)
2. {
3. tabControl1.SelectedTab = tabControl1.TabPages[1];
4. Graphics g;
5. **int** width = 100;
6. **int** height = 100;

9. image = **new** Bitmap(width, height);
10. g = Graphics.FromImage(image);
12. Color color = Color.FromArgb(255, 0, 0);//颜色为红色
13. Pen p = **new** Pen(color, 3);//创建一个画笔对象,该画笔的颜色为红色，笔触大小为3个像素
14. g.DrawEllipse(p, 50, 50, 100, 100);
16. pictureBox1.Image = (Image)image;
17. }

**10. tabcontrol中第二个界面（报告）加入TextBox控件。在Label控件属性面板中将AutoSize改为False，之后就可以控制文本框大小。将ScrollBars中的选项改为Vertical，将Multiline中的选项改为True，将WordWrap中的选项改为True。**

****

**此处负责结果显示**

**代码4：**

1. **private** **void** 报告ToolStripMenuItem\_Click(object sender, EventArgs e)
2. {
3. tabControl1.SelectedTab = tabControl1.TabPages[2];
4. textBox1.Text = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*坐标转换\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r\n";
5. textBox1.Text += "\n---------------BLH2XYZ---------------\r\n";
6. textBox1.Text += "ID \t B \t L \t H \t X \t Y \t Z \r\n";
7. **for**(**int** i = 0; i < s; i++)
8. {
9. textBox1.Text += point\_ID[i] + "\t" + point\_B[i] + "\t" + point\_L[i] + "\t" + point\_H[i] + "\t" + point\_X[i] + "\t" + point\_Y[i] + "\t" + point\_Z[i] + "\r\n";
10. }
12. }

**报告保存代码：**

1. **private** **void** 保存ToolStripMenuItem\_Click(object sender, EventArgs e)
2. {
3. **try**
4. {
5. saveFileDialog1.Title = "报告保存";
6. saveFileDialog1.Filter = "文本文档(\*.txt)|\*.txt";
7. **if** (saveFileDialog1.ShowDialog() == DialogResult.OK)
8. {
9. string path = saveFileDialog1.FileName;
10. **if** (textBox1.Text != "")
11. {
12. **using**(StreamWriter writer = **new** StreamWriter(path))
13. writer.Write(**this**.textBox1.Text);
14. }
15. MessageBox.Show("保存成功！");
16. }
17. }
18. **catch** (Exception ex)
19. {
20. MessageBox.Show(ex.Message);
21. **return**;
22. }
24. }

**图片保存代码：**

**需要全局bitmap类实例化对象**

1. **private** **void** 保存ToolStripMenuItem1\_Click(object sender, EventArgs e)
2. {
3. **try**
4. {
5. saveFileDialog1.Title = "bmp图形保存";
6. saveFileDialog1.Filter = "位图文件(\*.bmp)|\*.bmp";
7. **if** (saveFileDialog1.ShowDialog() == DialogResult.OK)
8. {
9. image.Save(saveFileDialog1.FileName);
10. MessageBox.Show("保存成功！");
11. }
12. }
13. **catch** (Exception ex)
14. {
15. MessageBox.Show(ex.Message);
16. **return**;
17. }
18. }

**算法部分代码：**

**仅供大概参考**

**全局变量：**

1. #region 定义变量
2. List<string> point\_ID = **new** List<string>();//创建一个库存放点的ID
3. List<**double**> point\_B = **new** List<**double**>();//创建一个库存放点的X
4. List<**double**> point\_L = **new** List<**double**>();//创建一个库存放点的Y
5. List<**double**> point\_H = **new** List<**double**>();//创建一个库存放点的Z
7. List<**double**> point\_X = **new** List<**double**>();//放X的库
8. List<**double**> point\_Y = **new** List<**double**>();//放Y的库
9. List<**double**> point\_Z = **new** List<**double**>();//放Z的库
11. List<**double**> point\_b = **new** List<**double**>();//放之后算出来的B的库
12. List<**double**> point\_l = **new** List<**double**>();//放之后算出来的L的库
13. List<**double**> point\_h = **new** List<**double**>();//放之后算出来的H的库
15. List<**double**> length = **new** List<**double**>();//放中央弧长
17. List<**double**> Gauss\_x = **new** List<**double**>();//放高斯投影x
18. List<**double**> Gauss\_y = **new** List<**double**>();//放高斯投影y
20. List<**double**> latitude = **new** List<**double**>();//放底点纬度
22. List<**double**> Gauss\_B = **new** List<**double**>();
23. List<**double**> Gauss\_L = **new** List<**double**>();
24. **double** a;
25. **double** f;
26. **double** L0;
27. **double** b;
28. **double** e1, e2;//第一偏心率，第二偏心率
29. **double** M0;
30. **int** s;
31. Bitmap image;
32. #endregion

**计算：**

1. **private** **void** 计算ToolStripMenuItem1\_Click(object sender, EventArgs e)
2. {
3. **int** z = 0;
4. **for** (**int** i = 0; i < s; i++)//BLH转XYZ转BLH
5. {
7. **double** W, n, t, N, M;
9. **double** sinB, cosB, tanB;
10. sinB = Math.Round(Math.Sin(point\_B[z] \* 1.00 / 180 \* Math.PI), 5);//sinB
11. cosB = Math.Round(Math.Cos(point\_B[z] \* 1.00 / 180 \* Math.PI), 5);//cosB
12. tanB = Math.Round(Math.Tan(point\_H[z] \* 1.00 / 180 \* Math.PI), 5);//tanB
14. W = Math.Sqrt(1 - (e1 \* e1) \* (sinB) \* sinB);
15. n = e2 \* cosB;
16. t = tanB;
18. N = a / W;//卯酉圈的曲率半径
19. M = (a \* (1 - (e1 \* e1)) / (W \* W \* W));//子午圈曲率半径
21. **double** sinL, cosL;
22. sinL = Math.Round(Math.Sin(point\_L[z] \* 1.00 / 180 \* Math.PI), 5);
23. cosL = Math.Round(Math.Cos(point\_L[z] \* 1.00 / 180 \* Math.PI), 5);
25. point\_X.Add((N + point\_H[z]) \* cosB \* cosL);//B转换为X
26. point\_Y.Add((N + point\_H[z]) \* cosB \* sinL);//L转换为Y
27. point\_Z.Add((N \* (1 - (e1 \* e1)) + point\_H[z]) \* sinB);//H转换为Z



32. dataGridView1.Rows[i].Cells[0].Value = point\_ID[i];
33. dataGridView1.Rows[i].Cells[1].Value = point\_B[i];
34. dataGridView1.Rows[i].Cells[2].Value = point\_L[i];
35. dataGridView1.Rows[i].Cells[3].Value = point\_H[i];
36. dataGridView1.Rows[i].Cells[4].Value = point\_X[i];
37. dataGridView1.Rows[i].Cells[5].Value = point\_Y[i];
38. dataGridView1.Rows[i].Cells[6].Value = point\_Z[i];
40. **double** X = point\_X[z] + 1000;//X+1000
41. **double** Y = point\_Y[z] + 1000;//Y+1000
42. **double** Z = point\_Z[z] + 1000;//Z+1000
44. **double** tan = Math.Sqrt((Z + Z) + ((Math.Sqrt((X \* X) + (Y \* Y))) \* (Math.Sqrt((X \* X) + (Y \* Y)))));//求斜边
45. sinB = (Z) / tan;//求sinB
46. **double** B1 = Math.Asin(sinB) / Math.PI \* 180;//将sinB中的B作为第一个初始的B1
47. **double** B2;
49. **while** (**true**)//迭代计算 根据B1代入公式算出B2 如果B2达标则跳出循环 如果不达标则将B2算作B1 继续计算
50. {
51. **double** W1 = Math.Sqrt(1 - (e1 \* e1) \* (sinB) \* sinB);
52. **double** N1 = a / W;
53. B2 = (Math.Atan(((point\_Z[z] + 1000) + (N1 \* e1 \* sinB)) / (Math.Sqrt(((X) \* (X)) + ((Y) \* (Y)))))) / Math.PI \* 180;
55. **if** (Math.Abs(B2 - B1) <= 0.0000000001)//如果差值小于它则跳出循环
56. {
57. **break**;
58. }
59. B1 = B2;
60. }
62. point\_l.Add((Math.Atan(Y / X)) / Math.PI \* 180);//将算出的L添加入库
63. point\_b.Add(B2);//将B添加入库
64. point\_h.Add(((Math.Sqrt((X \* X) + (Y \* Y))) / (Math.Round(Math.Cos(point\_b[z] \* 1.00 / 180 \* Math.PI), 5))) - N);//将H添加入库
66. //高斯正算
68. //子午弧长计算准备
69. **double** Ac, Bc, Cc, Dc, Ec, Fc;
71. Ac = 1 + ((3 / 4) \* e1) + ((45 / 64) \* Math.Pow(e1, 2)) + ((175 / 256) \* Math.Pow(e1, 3)) + ((11025 / 16384) \* Math.Pow(e1, 4)) + ((43659 / 65536) \* Math.Pow(e1, 5));
72. Bc = ((3 / 4) \* e1) + ((15 / 16) \* Math.Pow(e1, 2)) + ((252 / 512) \* Math.Pow(e1, 3)) + ((2205 / 2048) \* Math.Pow(e1, 4)) + ((72765 / 65536) \* Math.Pow(e1, 5));
73. Cc = ((15 / 64) \* Math.Pow(e1, 2)) + ((105 / 256) \* Math.Pow(e1, 3)) + ((2205 / 4096) \* Math.Pow(e1, 4)) + ((10395 / 16384) \* Math.Pow(e1, 5));
74. Dc = ((35 / 512) \* Math.Pow(e1, 3)) + ((315 / 2048) \* Math.Pow(e1, 4)) + ((31185 / 131072) \* Math.Pow(e1, 5));
75. Ec = ((315 / 16384) \* Math.Pow(e1, 4)) + ((3465 / 65536) \* Math.Pow(e1, 5));
76. Fc = ((693 / 131072) \* Math.Pow(e1, 5));
78. **double** alpha, beta, gamma, delta, epsilon, zeta;
79. alpha = Ac \* M0;
80. beta = -1 / 2 \* Bc \* M0;
81. gamma = 1 / 4 \* Cc \* M0;
82. delta = -1 / 6 \* Dc \* M0;
83. epsilon = 1 / 8 \* Ec \* M0;
84. zeta = -1 / 10 \* Fc \* M0;
86. //子午弧长计算
88. **double** sin2B, sin4B, sin6B, sin8B, sin10B;
89. sin2B = Math.Round(Math.Sin(point\_B[z] \* 2.00 / 180 \* Math.PI), 5);
90. sin4B = Math.Round(Math.Sin(point\_B[z] \* 4.00 / 180 \* Math.PI), 5);
91. sin6B = Math.Round(Math.Sin(point\_B[z] \* 6.00 / 180 \* Math.PI), 5);
92. sin8B = Math.Round(Math.Sin(point\_B[z] \* 8.00 / 180 \* Math.PI), 5);
93. sin10B = Math.Round(Math.Sin(point\_B[z] \* 10.00 / 180 \* Math.PI), 5);
94. length.Add(alpha \* point\_B[z] + beta \* sin2B + gamma \* sin4B + delta \* sin6B + epsilon \* sin8B + zeta \* sin10B);
96. //经差计算准备
97. **double** l = point\_L[z] - L0;
99. //计算辅助量
100. **double** alpha0, alpha1, alpha2, alpha3, alpha4, alpha5, alpha6;
101. alpha0 = length[z];
102. alpha1 = N \* cosB;
103. alpha2 = 1 / 2 \* N \* Math.Pow(cosB, 2) \* t;
104. alpha3 = 1 / 6 \* N \* Math.Pow(cosB, 3) \* (1 - t \* t + n \* n);
105. alpha4 = 1 / 24 \* N \* Math.Pow(cosB, 4) \* (5 - t \* t + 9 \* n \* n + 4 \* n \* n \* n \* n) \* t;
106. alpha5 = 1 / 120 \* N \* Math.Pow(cosB, 5) \* (5 - 18 \* t \* t + t \* t \* t \* t + 15 \* n \* n - 58 \* n \* n \* t \* t);
107. alpha6 = 1 / 720 \* N \* Math.Pow(cosB, 6) \* (61 - 58 \* t \* t + t \* t \* t \* t + 270 \* n \* n - 330 \* n \* n \* t \* t) \* t;
109. //高斯正算公式
110. Gauss\_x.Add(alpha0 + alpha2 \* l \* l + alpha \* l \* l \* l \* l + alpha6 \* Math.Pow(l, 6));
111. Gauss\_y.Add((alpha1 \* l + alpha3 \* l \* l \* l + alpha5 \* Math.Pow(l, 5)) + 500);

114. //高斯反算
116. //计算底点纬度
117. **double** Bf;
118. **double** XX, YY;
119. XX = Gauss\_x[z] + 1000;
120. YY = Gauss\_y[z] + 1000;
121. **while** (**true**)//迭代计算求底点纬度
122. {
123. **double** B0 = XX / alpha;
124. **double** sin2B0, sin4B0, sin6B0, sin8B0, sin10B0;
125. sin2B0 = Math.Round(Math.Sin(B0 \* 2.00 / 180 \* Math.PI), 5);
126. sin4B0 = Math.Round(Math.Sin(B0 \* 4.00 / 180 \* Math.PI), 5);
127. sin6B0 = Math.Round(Math.Sin(B0 \* 6.00 / 180 \* Math.PI), 5);
128. sin8B0 = Math.Round(Math.Sin(B0 \* 8.00 / 180 \* Math.PI), 5);
129. sin10B0 = Math.Round(Math.Sin(B0 \* 10.00 / 180 \* Math.PI), 5);
130. **double** triangle = beta \* sin2B0 + gamma \* sin4B0 + delta \* sin6B0 + epsilon \* sin8B0 + zeta \* sin10B0;
131. Bf = (XX - triangle) / alpha;
133. **if** (Math.Abs(Bf - B0) < 0.00000001)
134. {
135. **break**;
136. }
137. B0 = Bf;
138. }
140. latitude.Add(Bf);//将底点纬度放入库
142. //计算辅助量
143. **double** Wf, Nf, nf, Mf, tf;
144. **double** sinBf, cosBf, tanBf;
145. sinBf = Math.Round(Math.Sin(latitude[z] \* 1.00 / 180 \* Math.PI), 5);//sinBf
146. cosBf = Math.Round(Math.Cos(latitude[z] \* 1.00 / 180 \* Math.PI), 5);//cosBf
147. tanBf = Math.Round(Math.Tan(latitude[z] \* 1.00 / 180 \* Math.PI), 5);//tanBf
149. Wf = Math.Sqrt(1 - (e1 \* e1) \* (sinBf) \* sinBf);//Wf
150. nf = e2 \* cosBf;//nf
151. tf = tanBf;//tf
152. Nf = a / Wf;//卯酉圈的曲率半径
153. Mf = (a \* (1 - (e1 \* e1)) / (Wf \* Wf \* Wf));//子午圈曲率半径
155. **double** b0, b1, b2, b3, b4, b5, b6;
156. b0 = latitude[z];
157. b1 = 1 / (Nf \* cosBf);
158. b2 = -tf / (2 \* Mf \* Nf);
159. b3 = -((1 + (2 \* tf \* tf) + (nf \* nf)) / 6 \* (Nf \* Nf)) \* b1;
160. b4 = -((5 + (3 \* tf \* tf + nf \* nf) - (9 \* nf \* nf \* tf \* tf)) / (12 \* Nf \* Nf)) \* b2;
161. b5 = -((5 + (28 \* tf \* tf) + (24 \* tf \* tf \* tf \* tf) + (6 \* nf \* nf) + (8 \* nf \* nf \* tf \* tf)) / (120 \* Nf \* Nf \* Nf \* Nf)) \* b1;
162. b6 = ((61 + (90 \* tf \* tf) + (45 \* tf \* tf \* tf \* tf)) / (360 \* Nf \* Nf \* Nf \* Nf)) \* b2;
164. //高斯反算公式
165. Gauss\_B.Add(b0 + b2 \* YY \* YY + b4 \* YY \* YY \* YY \* YY + b6 \* Math.Pow(YY, 6));
166. Gauss\_L.Add(b1 \* YY + b3 \* YY \* YY \* YY + b5 \* Math.Pow(YY, 5) + L0);
168. z++;
169. }
170. }

**二.算法可能遇到的问题**

**1.角度转弧度时，转换一定要精确，否则结果会有细小偏差。**

**2.在使用除法时要在分母或者分子添加小数点后几位，否则结果会被自动四舍五入导致结果偏差。**

**三．声明**

**1.初始算法因不可抗力，未生成数据类，难以改写，遂算法未进行拆分，请自行改进C#代码结构，下有c++代码参考**

**2.结果精确度受弧度角度转换精度影响很大，要重视基本数据处理**

**Algo.h代码：**

**3.数据修改后的数据改变方面因算法结构问题未实现，下一题会给出示范**

1. #pragma once
2. #include<iostream>
3. #include<string>
4. #include<fstream>
5. #include<sstream>
6. #include<vector>
7. #include<math.h>
8. //using namespace std;
9. #define  PI 3.1415926
11. **struct** Data
12. {
13. **double** a;
14. **double** b;
15. **double** f;
16. **double** L0;
17. **double** e1;
18. **double** e2;
19. **double** M0;
20. };
21. **struct** Point
22. {
23. std::string ID;
24. **double** X, Y, Z;
25. **double** B, L, H;
26. };
27. **struct** Gauss
28. {
29. std::string ID;
30. **double** x, y;
31. **double** B, L;
32. };
34. **struct** Error
35. {
36. **int** errortag = 0;
37. };
39. **class** Calculate
40. {
41. **public**:
42. Data Dsrc;
43. std::vector<Point> Psrc;
44. std::vector<Point> Pdst;
45. std::vector<Gauss> GBL2xy;
46. std::vector<Gauss> Gxy2BL;
47. Error error;


51. **void** Get\_Data(std::string filepath);
52. **void** Algorithm();
53. **void** BLH2XYZ();
54. **void** XYZ2BLH();
55. **void** BL2xy();
56. **void** xy2BL();
57. };

**Algo.cpp代码：**

1. #include"Algo.h"
2. **using** **namespace** std;
3. **double** Angel2Rad(**double** dms)
4. {
5. **double** d = **int**(dms);
6. **double** m = **int**((dms - d) \* 100);
7. **double** s = 10000 \* (dms - d - m / 100);
8. **double** temp = d + m / 60 + s / 3600;
9. **double** rad = temp \* PI / 180;
10. **return** rad;
11. }
12. **double** Rad2Angel(**double** rad)
13. {
14. **double** du = rad \* 180 / PI;
15. **return** du;
16. }
17. **void** Calculate::Get\_Data(std::string filepath)
18. {
19. std::ifstream ifs(filepath);
20. ofstream ofs("Predata.txt");
21. string Tline;
22. string rubbish;
23. **if** (!ifs)
24. {
25. error.errortag = 1;
26. }
28. **while** (getline(ifs, Tline))
29. {
30. **for** (**int** i = 0; i < Tline.size(); i++)
31. {
32. **if**(Tline[i] == ',')
33. Tline[i] = ' ';
34. }
35. ofs << Tline << endl;
36. }
37. ofs.close(); ifs.close();
38. ifs.open("Predata.txt");
39. getline(ifs,Tline);
40. stringstream ss;
41. ss << Tline;
42. ss >> rubbish >> Dsrc.a;
43. ss.clear();
44. getline(ifs, Tline);
45. ss << Tline;
46. ss >> rubbish >> Dsrc.f;
47. ss.clear();
48. getline(ifs, Tline);
49. ss << Tline;
50. ss >> rubbish >> Dsrc.L0;
51. ss.clear();
52. Point p;
53. **while** (getline(ifs, Tline))
54. {
55. **if** (Tline.size() != 0)
56. {
57. ss << Tline;
58. ss >> p.ID >> p.B >> p.L >> p.H;
59. Psrc.push\_back(p);
60. ss.clear();
61. }
63. }
64. }
65. **void** Calculate::Algorithm()
66. {
67. Dsrc.f = 1/Dsrc.f;
68. Dsrc.b = Dsrc.a - Dsrc.a \* Dsrc.f;
69. **double** a = Dsrc.a;
70. **double** b = Dsrc.b;
71. **double** f = Dsrc.f;
72. **double** e1 = (a\*a -b\*b)/(a\*a);
73. **double** e2 = e1 / (1-e1);
74. **double** M0 = a\*(1 - e1);
75. Dsrc.e1 = e1;
76. Dsrc.e2 = e2;
77. Dsrc.M0 = M0;
78. /\*double w = sqrt(1 - e1 \* )\*/
79. **int** aaa = 0;
80. }
82. **void** Calculate::BLH2XYZ()
83. {
84. **for** (**int** i = 0; i < Psrc.size(); i++)
85. {
86. **double** B = Angel2Rad(Psrc[i].B);
87. **double** L = Angel2Rad(Psrc[i].L);
88. **double** H = Angel2Rad(Psrc[i].H);
90. **double** sinB = sin(B);
91. **double** cosB = cos(B);
92. **double** tanB = tan(B);
93. **double** cosL = cos(L);
94. **double** sinL = sin(L);
96. **double** W = sqrt(1 - Dsrc.e1 \* sinB\*sinB);
97. **double** N = Dsrc.a / W;

100. **double** X = (N + H) \* cosB \* cosL;
101. **double** Y = (N + H) \* cosB \* sinL;
102. **double** Z = (N \* (1 - Dsrc.e1) + H) \* sinB;
104. Psrc[i].X = X;
105. Psrc[i].Y = Y;
106. Psrc[i].Z = Z;

109. }
111. }
113. **void** Calculate::XYZ2BLH()
114. {
115. Pdst = Psrc;
116. **for** (**int** i = 0; i < Psrc.size(); i++)
117. {
118. Pdst[i].X = Psrc[i].X + 1000;
119. Pdst[i].Y = Psrc[i].Y + 1000;
120. Pdst[i].Z = Psrc[i].Z + 1000;
122. **double** X = Pdst[i].X;
123. **double** Y = Pdst[i].Y;
124. **double** Z = Pdst[i].Z;
126. //double eta = sqrt(sqrt(e2 \* cosB));
127. **double** r = sqrt(X\*X + Y\*Y);
128. **double** L = atan(Y / X);
129. //double Ldu = Rad2Angel(L);
130. **if** (X < 0)
131. L += PI;
133. **double** B1 = atan(Z / r);
134. **double** B2;
136. **while** (**true**)
137. {
138. **double** T\_W = sqrt(1 - (Dsrc.e1 \* sin(B1) \* sin(B1)));
139. **double** T\_N = Dsrc.a / T\_W;
140. B2 = atan((Z + T\_N\*Dsrc.e1\*sin(B1)) / r);
141. **if** (abs(B2 - B1) <= 1e-15)
142. **break**;
143. B1 = B2;
144. }
146. **double** W = sqrt(1 - Dsrc.e1 \* sin(B2)\*sin(B2));
147. **double** N = Dsrc.a / W;
148. **double** H = r / cos(B2) - N;
150. Pdst[i].B = B2;
151. Pdst[i].L = L;
152. Pdst[i].H = H;
153. }
154. }
156. **void** Calculate::BL2xy()
157. {
158. **double** M0 = Dsrc.M0;
159. **double** e1 = Dsrc.e1;
160. **double** e2 = Dsrc.e2;
162. **double** Ac = 1 + 3 \* e1 / 4.0 + 45 \* e1\*e1 / 64.0 + 175 \* pow(e1, 3) / 256.0 + 11025 \* pow(e1, 4) / 16384.0
163. + 43659 \* pow(e1, 5) / 65536.0;
164. **double** Bc = 3 \* e1 / 4.0 + 15 \* e1 \* e1 / 16.0 + 525 \* pow(e1, 3.0) / 512.0
165. + 2205 \* pow(e1, 4.0) / 2048.0 + 72765 \* pow(e1, 5.0) / 65536;
166. **double** Cc = 15 \* e1 \* e1 / 64.0 + 105 \* pow(e1, 3.0) / 256.0
167. + 2205 \* pow(e1, 4.0) / 4096.0 + 10395 \* pow(e1, 5.0) / 16384;
168. **double** Dc = 35 \* pow(e1, 3.0) / 512.0 + 315 \* pow(e1, 4.0) / 2048.0
169. + 31185 \* pow(e1, 5.0) / 131072;
170. **double** Ec = 315 \* pow(e1, 4.0) / 16384.0 + 3465 \* pow(e1, 5.0) / 65536;
171. **double** Fc = 693 \* pow(e1, 5.0) / 131072;
172. **double** alpha = Ac \* M0;
173. **double** beta = -Bc \* M0 / 2.0;
174. **double** gamma = Cc \* M0 / 4.0;
175. **double** sigma = -Dc \* M0 / 6.0;
176. **double** epsilon = Ec \* M0 / 8.0;
177. **double** zeta = -Fc \* M0 / 10.0;
179. **for** (**int** i = 0; i < Pdst.size(); i++)
180. {
182. **double** B = Pdst[i].B;
183. **double** L = Pdst[i].L;
184. **double** X = alpha \* B + beta \* sin(2 \* B) + gamma \* sin(4 \* B) +
185. sigma \* sin(6 \* B) + epsilon \* sin(8 \* B) + zeta \* sin(10 \* B);
186. **double** l = L - Angel2Rad(Dsrc.L0);
188. **double** W = sqrt(1 - e1 \* sin(B) \* sin(B));
189. **double** N = Dsrc.a / W;
190. **double** t = tan(B);
191. **double** eta2 = e2\*cos(B)\*cos(B);
192. //计算辅助量
193. **double** a0 = X;
194. **double** a1 = N \* cos(B);
195. **double** a2 = N \* cos(B) \* cos(B) \* t / 2.0;
196. **double** a3 = N \* pow(cos(B), 3.0) \* (1 - t \* t + eta2) / 6.0;
197. **double** a4 = N \* pow(cos(B), 4.0) \* (5 - t \* t + 9 \* eta2 + 24 \* eta2 \* eta2) \* t / 24.0;
198. **double** a5 = N \* pow(cos(B), 5.0) \* (5 - 18 \* t \* t + pow(t, 4.0) + 14 \* eta2 - 58 \* eta2 \* t \* t) / 120.0;
199. **double** a6 = N \* pow(cos(B), 6.0) \* (61 - 58 \* t \* t + pow(t, 4.0) + 270 \* eta2 - 330 \* eta2 \* t \* t) \* t / 720.0;
200. **double** x = a0 + a2 \* l \* l + a4 \* pow(l, 4.0) + a6 \* pow(l, 6.0);
201. **double** y = a1 \* l + a3 \* pow(l, 3.0) + a5 \* pow(l, 5.0);
202. y += 500000.0;
203. Gauss g;
204. g.ID = Pdst[i].ID;
205. g.L = Pdst[i].L;
206. g.B = Pdst[i].B;
207. g.x = x;
208. g.y = y;
209. GBL2xy.push\_back(g);
211. }
213. }
215. **void** Calculate::xy2BL()
216. {
217. **double** M0 = Dsrc.M0;
218. **double** e1 = Dsrc.e1;
219. **double** e2 = Dsrc.e2;
220. **double** L0 = Dsrc.L0 \* 3600.0 / 206265.0;
222. **double** Ac = 1 + 3 \* e1 / 4.0 + 45 \* e1\*e1 / 64.0 + 175 \* pow(e1, 3) / 256.0 + 11025 \* pow(e1, 4) / 16384.0
223. + 43659 \* pow(e1, 5) / 65536.0;
224. **double** Bc = 3 \* e1 / 4.0 + 15 \* e1 \* e1 / 16.0 + 525 \* pow(e1, 3.0) / 512.0
225. + 2205 \* pow(e1, 4.0) / 2048.0 + 72765 \* pow(e1, 5.0) / 65536;
226. **double** Cc = 15 \* e1 \* e1 / 64.0 + 105 \* pow(e1, 3.0) / 256.0
227. + 2205 \* pow(e1, 4.0) / 4096.0 + 10395 \* pow(e1, 5.0) / 16384;
228. **double** Dc = 35 \* pow(e1, 3.0) / 512.0 + 315 \* pow(e1, 4.0) / 2048.0
229. + 31185 \* pow(e1, 5.0) / 131072;
230. **double** Ec = 315 \* pow(e1, 4.0) / 16384.0 + 3465 \* pow(e1, 5.0) / 65536;
231. **double** Fc = 693 \* pow(e1, 5.0) / 131072;
232. **double** alpha = Ac \* M0;
233. **double** beta = -Bc \* M0 / 2.0;
234. **double** gamma = Cc \* M0 / 4.0;
235. **double** sigma = -Dc \* M0 / 6.0;
236. **double** epsilon = Ec \* M0 / 8.0;
237. **double** zeta = -Fc \* M0 / 10.0;
239. **int** count = 0;
241. **for** (**int** i = 0; i < GBL2xy.size(); i++)
242. {
243. **double** x = GBL2xy[i].x + 1000.0;
244. **double** y = GBL2xy[i].y + 1000.0;
245. **double** X = GBL2xy[i].x;
246. **double** B0 = X / alpha;
247. **double** Bf;
248. **double** delta = 0;
249. **while** (**true**)
250. {
251. Bf = (X - delta) / alpha;
252. delta = beta \* sin(2 \* B0) + gamma \* sin(4 \* B0) +
253. sigma \* sin(6 \* B0) + epsilon \* sin(8 \* B0) + zeta \* sin(10 \* B0);
254. **if** (abs(Bf - B0) < 0.00000001 && abs(Bf - B0) != 0)
255. **break**;
256. B0 = Bf;
257. }
258. //计算辅助量
259. **double** aa = cos(Bf);
260. **double** W = sqrt(1 - e1 \* sin(Bf) \* sin(Bf));
261. **double** N = Dsrc.a / W;
262. **double** t = tan(Bf);
263. **double** eta2 = e2\*cos(Bf)\*cos(Bf);
264. **double** M = N \* (1 - e1) / (W \* W \* W);
266. **double** b0 = Bf;
267. **double** b1 = 1 / (N \* cos(Bf));
268. **double** b2 = -t / (2 \* M \* N);
269. **double** b3 = -(1 + 2 \* t \* t + eta2) \* b1 / (6 \* N \* N);
270. **double** b4 = -(5 + 3 \* t \* t + eta2 - 9 \* eta2 \* t \* t) \* b2 / (12 \* N \* N);
271. **double** b5 = -(5 + 28 \* t \* t + 24 \* pow(t, 4.0) + 6 \* eta2 + 8 \* eta2 \* t \* t) \* b1 / (120 \* N \* N \* N \* N);
272. **double** b6 = (61 + 90 \* t \* t + 45 \* pow(t, 4.0)) \* b2 / (360 \* pow(N, 4.0));
274. **double** B = Rad2Angel(b0 + b2 \* y \* y + b4 \* pow(y, 4.0) + b6 \* (y, 6.0));
275. **double** L = Rad2Angel(b1 \* y + b3 \* pow(y, 3.0) + b5 \* pow(y, 5.0) + L0);
276. **int** aaa = 0;
277. }


281. }
283. **int** main()
284. {
285. std::string filepath = "坐标转换.txt";
286. Calculate Cal;
287. Cal.Get\_Data(filepath);
288. Cal.Algorithm();
289. Cal.BLH2XYZ();
290. Cal.XYZ2BLH();
291. Cal.BL2xy();
292. Cal.xy2BL();
293. getchar();
294. }