**文件类File**

using System.Windows.Forms;

using System.IO;

//获取文件路径

public string OpenPath()

{

string path = null;

OpenFileDialog fd = new OpenFileDialog();

fd.Filter = "txt 文件|\*.txt";

if (fd.ShowDialog() == DialogResult.OK)

path = fd.FileName;

return path;

}

//获取保存路径

public string SavePath(string F)

{

string path = null;

SaveFileDialog fd = new SaveFileDialog();

fd.Filter = F; //在主窗口中决定Filter a="XXX 文件|\*.XXX"

if (fd.ShowDialog() == DialogResult.OK)

path = fd.FileName;

return path;

}

//读取数据

public void ReadFile(List<方法名>实例的变量名)

{

StreamReader sr = new StreamReader(OpenPath());

string line = null;

int i = 0;

while ((line = sr.ReadLine()) != null)

{

方法名 p = new 方法名();

p.X = Convert.ToDouble(line.Split(',')[0]);

//...

实例的变量名.Add(p);

i++;

}

sr.Close();

}

//保存为TXT文件

public void SaveTXT(string str,string F)

{

StreamWriter sw = new StreamWriter(SavePath(F));

sw.Write(str);

sw.Close();

}

//保存BMP文件直接在主函数中添加即可

//public void SaveBMP()

//保存为DXF文件

public void SaveDXF(List<方法名> 实例的变量名,string F)

{

StreamWriter sw = new StreamWriter(SavePath(F));

sw.WriteLine("0");

sw.WriteLine("SECTION");

sw.WriteLine("2");

sw.WriteLine("ENTITIES");

//点层

for(int i = 0; i < 实例的变量名.Count; i++)

{

sw.WriteLine("0");

sw.WriteLine("POINT");

sw.WriteLine("8");

sw.WriteLine("点层");

sw.WriteLine("10");

sw.WriteLine(实例的变量名[i].X); //X坐标值的输入

sw.WriteLine("20");

sw.WriteLine(实例的变量名[i].Y); //Y坐标值的输入

}

//线层

for (int i = 0; i < 实例的变量名.Count - 1; ++i)

{

sw.WriteLine("0");

sw.WriteLine("LINE");

sw.WriteLine("8");

sw.WriteLine("线层");

sw.WriteLine("10"); // 起点X坐标

sw.WriteLine(实例的变量名[i].X);

sw.WriteLine("20"); // 起点Y坐标

sw.WriteLine(实例的变量名[i].Y);

sw.WriteLine("11"); // 终点X坐标

sw.WriteLine(实例的变量名[i + 1].X);

sw.WriteLine("21"); // 终点Y坐标

sw.WriteLine(实例的变量名[i + 1].Y);

}

sw.WriteLine("0");

sw.WriteLine("ENDSEC");

sw.WriteLine("0");

sw.WriteLine("EOF");

**显示类Display**

using System.Data;

using System.Windows.Forms.DataVisualization.Charting;

using System.Drawing;

//刷新数据表格

public DataTable DataGridView(List<方法名> 实例的变量名)

{

DataTable dt = new DataTable();

dt.Columns.Add("...");

//......

dt.Columns.Add("...");

for(int i = 0; i < 实例的变量名.Count; i++)

{

DataRow dr = dt.NewRow();

dr[0] = 实例的变量名[i].XXX;

//......

dr[100] = 实例的变量名[i].XXX;

dt.Rows.Add(dr);

}

return dt;

}

//刷新图形界面

public void Chart(Chart chart, List<方法名> 实例的变量名)

{

//三清初始化

chart.Legends.Clear();

chart.Annotations.Clear();

chart.ChartAreas.Clear();

//创建新的ChartAreas实例并初始化各项参数

ChartArea ca = new ChartArea();

ca.AxisX.LineColor = Color.Black;

ca.AxisX.LineWidth = 3;

ca.AxisX.Title = "XXX";

ca.AxisX.MajorGrid.Enabled = false;

ca.AxisY.LineColor = Color.Black; //Y轴类似设置

//ca.AxisY...

//设置坐标范围

double MaxX = 实例的变量名[0].XX;

double MinX = 实例的变量名[0].XX;

//double MaxY;...

//double MinY;...

for(int i=0;i< 实例的变量名.Count; i++)

{

if (MaxX < 实例的变量名[i].XX) MaxX = 实例的变量名[i].XX;

if (MinX > 实例的变量名[i].XX) MinX = 实例的变量名[i].XX;

//...

//...

}

ca.AxisX.Maximum = MaxX + 150;

ca.AxisX.Minimum = MinX - 150;

//...

//...

//添加数据 点

chart.Series.Add("点");

chart.Series["点"].ChartType = SeriesChartType.Point;

chart.Series["点"].MarkerColor = Color.Black;

chart.Series["点"].MarkerSize = 5;

Series s1 = chart.Series["点"];

//添加数据 线

chart.Series.Add("线");

chart.Series["线"].ChartType = SeriesChartType.Line;

chart.Series["线"].Color = Color.Black;

chart.Series["线"].BorderWidth = 2;

Series s2 = chart.Series["线"];

//加载数据到chart

for(int i=0;i< 实例的变量名.Count; i++)

{

//加载点数据

s1.Points.AddXY(实例的变量名[i].X, 实例的变量名[i].Y);

//加载线数据

//注意是两点间连线，要连成一圈则需要在最后添加起始点

s2.Points.AddXY(实例的变量名[i].X, 实例的变量名[i].Y);

//加载备注

TextAnnotation t = new TextAnnotation();

t.AnchorDataPoint = s1.Points[i]; //获取数据点位置为注释定位

t.Text = 实例的变量名[i].XXX; //对应点的名称

chart.Annotations.Add(t);

}

chart.ChartAreas.Add(ca);

}

//刷新报告界面

public string Report(List<方法名> 实例的变量名, List<方法名> 实例的变量名2)

{

string str = null;

//添加说明信息

str += "...\r\n"

+ "...\r\n";

//导入数据

for(int i=0;i< 实例的变量名.Count; i++)

{

str += string.Format("{0,20:f4}{1,20:f4}...", "...", "..."); //{对应数据的下标，与下一个数据间的间隔，数据保留的位数（:f4为保留）}

}

return str;

}

**计算类Calculate**

//不需使用Math.Round来限制数据精度，保留精度由Display来控制显示

public void XXX(List<方法名> 实例的变量名,方法名 实例的变量名2)

{

//若最终输出的数据为一个“数组”，直接对该形参进行表达式运算即可

实例的变量名2.XXX = "表达式运算";

//实例的变量名2...

//若最终输出的数据为集合，先定义一个临时的同方法实例，用于进行计算，套上循环，每次循环结束后再用Add添加入输出的集合中

for (int i = 0; i < 实例的变量名.Count; i++)

{

方法名 p = new 方法名();

p.XXX = "表达式运算";

//...

p.XXX = "表达式运算";

实例的变量名.Add(p);

}

}

#region 其他计算

//角度转字符串格式,a为字符串形式的角度dd.mmssss

public string AngleTFStr(double a)

{

double du, fen, miao;

du = Math.Truncate(a);

fen = Math.Truncate((a - du) \* 100);

miao = Math.Truncate((a - du - fen) \* 10000);

string angle = du.ToString() + "°" + fen.ToString() + "′" + miao.ToString() + "″";

return angle;

}

//角度转双浮点格式,a为字符串形式的角度dd.mmssss

public double AngleTFDou(double a)

{

double du, fen, miao;

du = Math.Truncate(a);

fen = Math.Truncate((a - du) \* 100) / 60.0;

miao = Math.Truncate((a - du - fen) \* 10000) / 3600.0;

double angle = (du + fen + miao) / 180.0 \* Math.PI;

return angle;

}

#endregion

------10.16------

控制显示窗口的下标

tabControl1.SelectedIndex = 1;

绘图界面控制坐标轴上数值的显示位数

ca.AxisX.LabelStyle.Format = "f3";

ca.AxisY.LabelStyle.Format = "f3";

（显示小数点后3位）

图片拖动功能

