ContourPolyline.cs

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using System.Text;
5.  using System.Threading.Tasks;
6.
7.  namespace AGIS_work.DataStructure
8.  {
9.      public class ContourPolyline
10.     {
11.         public int PID { get; private set; }
12.         private static int _pid = 777777;
13.         public List<DataPoint> PointList = new List<DataPoint>();
14.         public ContourPolyline() { this.PID = _pid++; }
15.         public ContourPolyline(DataPoint[] points)
16.         {
17.             this.PointList.AddRange(points);
18.             this.PID = _pid++;
19.         }
20.         public static Object[] IntersectResult(ContourPolyline pl1, Edge edge)
21.         {
22.             List<ContourPolyline> sublineFromPL1 = new List<ContourPolyline>();
23.             List<Edge> suEdgeFromEdge = new List<Edge>();
24.             //对边上点排序
25.             List<DataPoint> subEdgePoint = new List<DataPoint>();
26.             subEdgePoint.Add(edge.StartPoint);
27.             subEdgePoint.Add(edge.EndPoint);
28.             edge.StartPoint.RelativeLoc = 0;
29.             edge.EndPoint.RelativeLoc = 1;
30.             //对折线上点排序
31.             List<DataPoint> subLinePoint = new List<DataPoint>();
32.             subLinePoint.Add(pl1.PointList[0]);
33.             for (int i = 0; i < pl1.PointList.Count - 1; i++)
34.             {
35.                 Edge pl1OneEdge = new Edge(pl1.PointList[i], pl1.PointList[i + 1]);
36.                 DataPoint intersectP = Edge.IntersectPoint(pl1OneEdge, edge);
37.                 double relativeLocOnLine = Edge.IntersectPointRelativeLoc(pl1OneEdge, edge);
38.                 double relativeLocOnEdge = Edge.IntersectPointRelativeLoc(edge, pl1OneEdge);
39.                 if (intersectP != null)
40.                 {
41.                     if (relativeLocOnEdge < 1 && relativeLocOnEdge > 0)
42.                     { intersectP.RelativeLoc = relativeLocOnEdge; subEdgePoint.Add(intersectP); }

43.                     if (relativeLocOnLine <= 1 && relativeLocOnLine > 0)
```

```
                    {
                        subLinePoint.Add(intersectP);
                        sublineFromPL1.Add(new ContourPolyline(subLinePoint.ToArray()));
                        subLinePoint = new List<DataPoint>();
                        subLinePoint.Add(intersectP);
                    }
                }
                subLinePoint.Add(pl1.PointList[i + 1]);
            }
            sublineFromPL1.Add(new ContourPolyline(subLinePoint.ToArray()));
            subEdgePoint.Sort((x, y) => x.RelativeLoc.CompareTo(y.RelativeLoc));
            for (int i = 0; i < subEdgePoint.Count - 1; i++)
            {
                suEdgeFromEdge.Add(new Edge(subEdgePoint[i], subEdgePoint[i + 1]));
            }
            return new Object[2] { sublineFromPL1, suEdgeFromEdge };
        }

        public static Object[] IntersectResult(ContourPolyline[] plineList, Edge edge)
        {
            List<ContourPolyline> sublineFromPLs = new List<ContourPolyline>();
            List<Edge> suEdgeFromEdge = new List<Edge>();
            //对边上点排序
            List<DataPoint> subEdgePoint = new List<DataPoint>();
            subEdgePoint.Add(edge.StartPoint);
            subEdgePoint.Add(edge.EndPoint);
            edge.StartPoint.RelativeLoc = 0;
            edge.EndPoint.RelativeLoc = 1;
            for (int k = 0; k < plineList.Length; k++)
            {
                //对折线上点排序
                List<DataPoint> subLinePoint = new List<DataPoint>();
                ContourPolyline curCpl = plineList[k];
                subLinePoint.Add(curCpl.PointList[0]);
                //选取一个等值线
                for (int i = 0; i < curCpl.PointList.Count - 1; i++)
                {
                    Edge pl1OneEdge = new Edge(curCpl.PointList[i], curCpl.PointList[i + 1]);
                    DataPoint intersectP = Edge.IntersectPoint(pl1OneEdge, edge);
                    double relativeLocOnLine = Edge.IntersectPointRelativeLoc(pl1OneEdge, edge);
                    double relativeLocOnEdge = Edge.IntersectPointRelativeLoc(edge, pl1OneEdge);
                    if (intersectP != null)
                    {
                        if (relativeLocOnEdge < 1 && relativeLocOnEdge > 0)
                        { intersectP.RelativeLoc = relativeLocOnEdge; subEdgePoint.Add(intersectP)
    ; }
                        if (relativeLocOnLine <= 1 && relativeLocOnLine > 0 )
                        {
```

```
                            if (subLinePoint.Count == 1 && subLinePoint[0].OID == intersectP.OID) { }
                            else
                            {
                                subLinePoint.Add(intersectP);
                                sublineFromPLs.Add(new ContourPolyline(subLinePoint.ToArray()));
                                subLinePoint = new List<DataPoint>();
                                subLinePoint.Add(intersectP);
                            }
                        }
                    }
                    subLinePoint.Add(curCpl.PointList[i + 1]);
                }
                if (subLinePoint.Count >= 2 && !(subLinePoint.Count == 2 && subLinePoint[0].OID == subLinePoint[1].OID))
                    sublineFromPLs.Add(new ContourPolyline(subLinePoint.ToArray()));
            }
            subEdgePoint.Sort((x, y) => x.RelativeLoc.CompareTo(y.RelativeLoc));
            for (int i = 0; i < subEdgePoint.Count - 1; i++)
            {
                if (subEdgePoint[i].RelativeLoc != subEdgePoint[i + 1].RelativeLoc)
                    suEdgeFromEdge.Add(new Edge(subEdgePoint[i], subEdgePoint[i + 1]));
            }
            return new Object[2] { sublineFromPLs, suEdgeFromEdge };
        }

        public override string ToString()
        {
            return string.Format("CLid:{0},PtsCount:{1}", this.PID, this.PointList.Count);
        }
    }
}
```