# DataPoint.cs

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Linq;
4. **using** System.Text;
5. **using** System.Threading.Tasks;
7. **namespace** AGIS\_work.DataStructure
8. {
9. **public** **class** DataPoint
10. {
11. **public** **int** ID { **get**; **private** **set**; }
12. **public** **string** Name { **get**; **private** **set**; }
13. **public** **double** X { **get**; **private** **set**; }
14. **public** **double** Y { **get**; **private** **set**; }
15. **public** **double** Value { **get**; **private** **set**; }
16. **public** MinBoundRect MBR { **get**; **private** **set**; }
17. **public** **int** OID { **get**; **private** **set**; }
18. **private** **static** **int** \_oid = 1000000;
19. **public** **double** RelativeLoc { **get**; **set**; }
21. **public** DataPoint(**int** id, **string** name, **double** x, **double** y, **double** value,**int** oid)
22. {
23. **this**.ID = id;
24. **this**.Name = name;
25. **this**.X = x;
26. **this**.Y = y;
27. **this**.Value = value;
28. **this**.MBR = **new** MinBoundRect(x, y, x, y);
29. **this**.OID = oid;
30. }
32. **public** DataPoint(**int** id, **string** name, **double** x, **double** y, **double** value)
33. {
34. **this**.ID = id;
35. **this**.Name = name;
36. **this**.X = x;
37. **this**.Y = y;
38. **this**.Value = value;
39. **this**.MBR = **new** MinBoundRect(x, y, x, y);
40. **this**.OID = \_oid++;
41. }
43. **public** **override** **string** ToString()
44. {
45. **return** **string**.Format("ID:{0} Name:{1}\r\n Point({2},{3})\r\nValue:{4}",
46. ID, Name, X, Y, Value);
47. }
49. //获取与另一点得距离
50. **public** **double** GetDistance(DataPoint other)
51. {
52. **return** Math.Sqrt(Math.Pow(**this**.X - other.X, 2) + Math.Pow(**this**.Y - other.Y, 2));
53. }
55. **public** **double** GetDistance(**double** x, **double** y)
56. {
57. **return** Math.Sqrt(Math.Pow(**this**.X - x, 2) + Math.Pow(**this**.Y - y, 2));
58. }
60. **public** **double** GetDistanceP2(**double** x, **double** y)
61. {
62. **return** (Math.Pow(**this**.X - x, 2) + Math.Pow(**this**.Y - y, 2));
63. }
65. //获取在另一点的方位角(角度)
66. **public** **double** GetPosition(**double** x,**double** y)
67. {
68. **double** deltaX = **this**.X - x;
69. **double** deltaY = **this**.Y - y;
70. **if** (deltaX \* deltaY == 0)
71. {
72. **if** (deltaX == 0)
73. {
74. **if** (deltaY > 0)
75. **return** 90;
76. **else** **if** (deltaY < 0)
77. **return** 270;
78. **else**
79. **throw** **new** Exception("DataPoint.GetPosition:两点重合");
80. }
81. **else**
82. {
83. **if** (deltaX > 0)
84. **return** 0;
85. **else** **return** 180;
86. }
87. }
88. **else**
89. {
90. **double** alpha = Math.Atan(Math.Abs(deltaY / deltaX));
91. **if** (deltaX > 0)
92. {
93. **if** (deltaY > 0) **return** alpha;
94. **else** **return** 360 - alpha;
95. }
96. **else**
97. {
98. **if** (deltaY > 0) **return** 180 - alpha;
99. **else** **return** 180 + alpha;
100. }
101. }
103. }
105. **public** **static** Vector2D **operator** - (DataPoint p1 ,DataPoint p2)
106. {
107. **return** **new** Vector2D(p1.X - p2.X, p1.Y - p2.Y);
108. }
110. **public** **static** **double** Angle(DataPoint c, DataPoint a, DataPoint b)
111. {
112. **double** ang;
113. **double** l1 = Math.Sqrt((b.X - c.X) \* (b.X - c.X) + (b.Y - c.Y) \* (b.Y - c.Y));
114. **double** l2 = Math.Sqrt((a.X - c.X) \* (a.X - c.X) + (a.Y - c.Y) \* (a.Y - c.Y));
115. **double** l3 = Math.Sqrt((b.X - a.X) \* (b.X - a.X) + (b.Y - a.Y) \* (b.Y - a.Y));
116. ang = Math.Acos((l1 \* l1 + l2 \* l2 - l3 \* l3) / (2 \* l1 \* l2));
117. **return** ang;
118. }
120. **public** **static** **int** LeftOrRight(DataPoint c, DataPoint a, DataPoint b)
121. {
122. **int** youbian;
123. **double** S;
124. S = (a.X - c.X) \* (b.Y - c.Y) - (a.Y - c.Y) \* (b.X - c.X);
125. **if** (S > 0)
126. {
127. youbian = 1;
128. }
129. **else** **if** (S < 0)
130. {
131. youbian = -1;
132. }
133. **else**
134. {
135. youbian = 0;
136. }
137. **return** youbian;
138. }
139. }
140. }