```
 1: /*
 2:  * AdjacencyList.cpp
 3:  *
 4:  *  Created on: May 28, 2011
 5:  *      Author: darioandrade
 6:  */
 7:
 8: #include "AdjacencyList.h"
 9: #include <cstdlib>
10:
11: #define MAX_LINE_SIZE ( 64 * 1024 )
12:
13: AdjacencyList::AdjacencyList ( )
14: :
15:   m_nVertex( 0 ),
16:   m_nEdges( 0 )
17: {
18: }
19:
20: AdjacencyList::AdjacencyList ( int nVertex )
21: {
22:     Allocate( nVertex );
23:
24:     m_nVertex = nVertex;
25: }
26:
27: AdjacencyList::~AdjacencyList ( )
28: {
29:     for(int i = 0; i < m_nVertex; i++) {
30:         delete m_arrAdjLists[i];
31:     }
32:
33:     delete m_arrAdjLists;
34: }
35:
36: void AdjacencyList::Allocate( int nVertex )
37: {
38:     m_arrAdjLists = new List * [ nVertex ];
39:
40:     for(int i = 0; i < nVertex; i++) {
41:         m_arrAdjLists[i] = new List();
42:     }
43: }
44:
45: void AdjacencyList::addEdge ( int iVertex, int jVertex, bool bUpdateNeighbor, bool bIncEdge )
46: {
47:     m_arrAdjLists[ iVertex ]->insertAtEnd( jVertex );
48:
49:     if ( bIncEdge )
50:     {
51:         m_nEdges ++;
52:     }
53:
54:     if ( bUpdateNeighbor )
55:     {
56:         m_arrAdjLists[ jVertex ]->insertAtEnd( iVertex );
57:     }
58: }
59:
60: // print edges, one vertex each line, to stdout
61: void AdjacencyList::write ( FILE * f )
62: {
63:     if ( f == NULL )
64:     {
65:         f = stdout;
66:     }
67:
68:     fprintf( f, "%d\n", m_nVertex );
69:
70:     // run all vertex
71:     for ( int i = 0; i < m_nVertex; i++ )
72:     {
73:
74:
75:         // iterate through edges
76:         for (ListNode * node = m_arrAdjLists[i]->getFirst(); node != NULL ; node = node->next())
77:         {
78:             fprintf( f, "%d ", node->getVertex() );
79:         }
80:
81:         fputs( "\n", f );
82:     }
83: }
84:
85: void AdjacencyList::read( FILE * f, int debug )
86: {
87:     if ( f == NULL )
88:     {
89:         f = stdin;
90:     }
91:
92:     static char sLine[ MAX_LINE_SIZE ];
93:
94:     fgets( sLine, MAX_LINE_SIZE, f );
```

```
 95:
 96:     m_nVertex = atoi( sLine );
 97:
 98:     Allocate( m_nVertex );
 99:
100:     // read all lines
101:     for ( int nCurrentVertex = 0;
102:             fgets( sLine, MAX_LINE_SIZE, f ) != NULL;
103:             nCurrentVertex++ )
104:     {
105:         if ( debug >= 2 )
106:         {
107:             fprintf( stderr, "\n  lendo linha do vertice %d:\n", nCurrentVertex );
108:         }
109:
110:         int offset = 0;
111:
112:         // read all neighbors from the line
113:         for ( int i = 0; i < MAX_LINE_SIZE; i++ )
114:         {
115:             int nNeighborVertex;
116:
117:             // only read if offset is within array boundaries
118:             if ( offset < MAX_LINE_SIZE )
119:             {
120:                 // read neighbor
121:                 int ret = sscanf( sLine + offset, " %d", &nNeighborVertex );
122:
123:                 // did we read a neighbor? or end of line/file?
124:                 if ( ret != EOF && ret > 0 )
125:                 {
126:                     if ( debug >= 2 )
127:                     {
128:                         fprintf( stderr, " %d ", nNeighborVertex );
129:                     }
130:
131:                     // file has vertex in ascending order, one each line
132:                     // if neighbor that is being added is greater, count the edge, otherwise
133:                     // it's already counted
134:                     addEdge( nCurrentVertex, nNeighborVertex, false, nNeighborVertex > nCurrentVertex );
135:
136:                     // find next item in line
137:                     while ( offset < MAX_LINE_SIZE
138:                             && sLine[ offset ] )
139:                     {
140:                         offset ++;
141:
142:                         if ( sLine[ offset ] == ' ' )
143:                         {
144:                             break;
145:                         }
146:                     }
147:                 }
148:                 else
149:                 {
150:                     // stop reading the line
151:                     break;
152:                 }
153:             }
154:         }
155:     }
156:
157:     // update other structures
158:     updateData();
159:
160:     if ( debug >= 2 )
161:     {
162:         fprintf( stderr, "\n fim do stream de leitura\n" );
163:     }
164: }
165:
166: bool AdjacencyList::HasEdge( ) const
167: {
168:     return m_nEdges > 0;
169: }
170:
```