

```

1  /*
2  * proj3
3  * Working example of a templated doubly linked list of type int.  Loads a specially formatted
4  * commands file which, when read by the program, constructs a doubly linked list with a particular
5  * configuration.  Commands are reported to the console.
6  *
7  * Programmer:  Chad Philip Johnson
8  * Date Created:  Thursday, March 28th, 2013
9  * Last Date Modified:  Sunday, April 14th, 2013
10 *
11 * Sources Used:
12 *     DLNode.h
13 *         - to create instances of individual nodes containing values of the supplied type within a doubly linked list
14 *     DLLList.h
15 *         - the doubly linked list apparatus
16 */
17
18 #pragma once
19
20 #include <cstdlib>
21 #include <string>
22 #include <iostream>
23 #include <fstream>
24
25 #include "DLNode.h"
26 #include "DLLList.h"
27
28 using namespace std;
29
30 /**
31 * Processes the program commands input from a formatted data file.
32 * @param pObjDLLListDriver The instance of the doubly linked list to be built or modified.
33 * @param charCommand The command to be issued to the doubly linked list.
34 * @param strInput The value to be added to the doubly linked list.
35 */
36 template<typename T>
37 void processInput( DLLList<T>*& pObjDLLListDriverList, char& charCommand, string strInput = "" );
38
39 int main( int argc, char* argv[] )
40 {
41     DLLList<int>* pObjDLLListDriverList    = NULL;
42
43     if( argc > 1 )
44     {
45         ifstream objifstreamInputFile( argv[1] );
46         if( objifstreamInputFile.good() )
47         {
48             string strNextLine  = "";
49
50             while( getline( objifstreamInputFile, strNextLine ) )

```

```

51     {
52         if( strNextLine.length() == 1 )
53         {
54             processInput( pObjDLListDriverList, strNextLine[0] );
55         }
56         else if( strNextLine.length() > 1 )
57         {
58             processInput( pObjDLListDriverList, strNextLine[0], strNextLine.substr( 2 ) );
59         }
60     }
61
62     objifstreamInputFile.close();
63 }
64 else
65 {
66     cout << "The file " << argv[1] << " does not exist." << endl;
67 }
68 }
69 else
70 {
71     cout << "Usage of proj3.exe:" << endl;
72     cout << "\t" << "proj3.exe NAME_OF_FILE" << endl;
73 }
74 }
75
76
77 template<class T>
78 void processInput( DLLList<T>*& pObjDLListDriverList, char& charCommand, string strInput )
79 {
80
81     switch( charCommand )
82     {
83         case '#':
84             break;
85
86         case 'C':
87         case 'c':
88             if( pObjDLListDriverList != NULL )
89             {
90                 delete pObjDLListDriverList;
91                 pObjDLListDriverList = NULL;
92             }
93             pObjDLListDriverList = new DLLList<int>;
94
95             cout << "LIST CREATED" << endl;
96             break;
97     }
98
99     if( pObjDLListDriverList != NULL )
100     {

```

```

101     switch( charCommand )
102     {
103         case 'X':
104         case 'x':
105             pObjDLListDriverList->clear();
106             cout << "LIST CLEARED" << endl;
107             break;
108
109         case 'D':
110         case 'd':
111             delete pObjDLListDriverList;
112             pObjDLListDriverList = NULL;
113             cout << "LIST DELETED" << endl;
114             break;
115
116         case 'I':
117         case 'i':
118             pObjDLListDriverList->insert( atoi( strInput.c_str() ) );
119             cout << "VALUE " << strInput << " INSERTED" << endl;
120             break;
121
122         case 'F':
123         case 'f':
124             pObjDLListDriverList->pushFront( atoi( strInput.c_str() ) );
125             cout << "VALUE " << strInput << " ADDED TO HEAD" << endl;
126             break;
127
128         case 'B':
129         case 'b':
130             pObjDLListDriverList->pushBack( atoi( strInput.c_str() ) );
131             cout << "VALUE " << strInput << " ADDED TO TAIL" << endl;
132             break;
133
134         case 'A':
135         case 'a':
136             try
137             {
138                 cout << "VALUE " << pObjDLListDriverList->getFront() << " AT HEAD" << endl;
139             }
140             catch( const char* e )
141             {
142                 cout << e << endl;
143             }
144             break;
145
146         case 'Z':
147         case 'z':
148             try
149             {
150                 cout << "VALUE " << pObjDLListDriverList->getBack() << " AT TAIL" << endl;

```

```

151     }
152     catch( const char* e )
153     {
154         cout << e << endl;
155     }
156     break;
157
158 case 'T':
159 case 't':
160     try
161     {
162         if( pObjDLListDriverList->getSize() == 0 )
163         {
164             throw "LIST EMPTY";
165         }
166         pObjDLListDriverList->popFront();
167         cout << "REMOVED HEAD" << endl;
168     }
169     catch( const char* e )
170     {
171         cout << e << endl;
172     }
173     break;
174
175 case 'K':
176 case 'k':
177     try
178     {
179         if( pObjDLListDriverList->getSize() == 0 )
180         {
181             throw "LIST EMPTY";
182         }
183         pObjDLListDriverList->popBack();
184         cout << "REMOVED TAIL" << endl;
185     }
186     catch( const char* e )
187     {
188         cout << e << endl;
189     }
190     break;
191
192 case 'E':
193 case 'e':
194     if( pObjDLListDriverList->removeAll( atoi( strInput.c_str() ) ) )
195     {
196         cout << "VALUE " << strInput << " ELIMINATED" << endl;
197     }
198     else
199     {
200         cout << "VALUE " << strInput << " NOT FOUND" << endl;

```

```

201     }
202     break;
203
204     case 'R':
205     case 'r':
206         if( pObjDLListDriverList->removeFirst( atoi( strInput.c_str() ) ) )
207         {
208             cout << "VALUE " << strInput << " REMOVED" << endl;
209         }
210         else
211         {
212             cout << "VALUE " << strInput << " NOT FOUND" << endl;
213         }
214         break;
215
216     case 'G':
217     case 'g':
218         if( pObjDLListDriverList->get( atoi( strInput.c_str() ) ) )
219         {
220             cout << "VALUE " << strInput << " FOUND" << endl;
221         }
222         else
223         {
224             cout << "VALUE " << strInput << " NOT FOUND" << endl;
225         }
226         break;
227
228     case 'N':
229     case 'n':
230         cout << "LIST SIZE IS " << pObjDLListDriverList->getSize() << endl;
231         break;
232
233     case 'P':
234     case 'p':
235         try
236         {
237             if( pObjDLListDriverList->getSize() == 0 )
238             {
239                 throw "LIST EMPTY";
240             }
241             cout << (*pObjDLListDriverList);
242         }
243         catch( const char* e )
244         {
245             cout << e << endl;
246         }
247         break;
248     }
249 }
250 else if( charCommand != '#' )

```

```
251     {  
252         cout << "MUST CREATE LIST INSTANCE" << endl;  
253     }  
254 }  
255
```