

```

1  #ifdef _WIN32
2      #include <tchar.h>
3      #include <conio.h>
4  #elif (defined __linux__) || (defined _AIX) || (defined __APPLE__)
5      typedef char _TCHAR;
6      #define _tmain main
7  #endif
8
9  #include <stdio.h>
10 #include <iostream>
11 using namespace std;
12
13 #include "LinkedList2.h"
14 //-----
15 TListNode::TListNode()
16 {
17     Data = 0;
18     nextPtr = NULL; // dolezite !!!!!!!!!!!!!!!!!!!!!!!
19 }
20 //-----
21 TListNode::~TListNode()
22 {
23 }
24 //-----
25 TList::TList()
26 {
27     firstPtr = lastPtr = NULL; // dolezite !!!!!!!!!!!!!!!!!!!!!!!
28 }
29 //-----
30 TList::~TList()
31 {
32     TListNode *currentPtr = firstPtr, *tempPtr;
33
34     if (!isEmpty()) {
35         cout << "\nDeallocating the nodes ...\n";
36
37         while (currentPtr != NULL) {
38             tempPtr = currentPtr;
39             cout << tempPtr->Data << endl;
40             currentPtr = currentPtr->nextPtr;
41             delete tempPtr;
42         }
43         cout << "All the nodes are deallocated.\n";
44     }
45 }
46 //-----
47 void TList::insertAtFront(const int &data)
48 {
49     TListNode *newPtr = new TListNode();
50
51     newPtr->Data = data;
52     if (isEmpty())
53         firstPtr = lastPtr = newPtr;
54     else {
55         newPtr->nextPtr = firstPtr;
56         firstPtr = newPtr;
57     }
58 }
59 //-----
60 void TList::insertBeforeElement(const int &data)
61 {
62     TListNode *newPtr = new TListNode();
63
64     newPtr->Data = data;
65     if (isEmpty())
66         firstPtr = lastPtr = newPtr;
67     else {
68         TListNode *currentPtr = firstPtr, *tempPtr = firstPtr;
69
70         if (data < currentPtr->Data) {
71             newPtr->nextPtr = firstPtr;
72             firstPtr = newPtr;
73         }
74     }
75 }

```

```

74     else {
75         while ((currentPtr != NULL) && (data >= currentPtr->Data)) {
76             tempPtr = currentPtr;
77             currentPtr = currentPtr->nextPtr;
78         }
79
80         newPtr->nextPtr = currentPtr;
81         tempPtr->nextPtr = newPtr;
82         if (currentPtr == NULL)
83             lastPtr = newPtr;
84     }
85 }
86 }
87 //-----
88 void TList::insertAtBack(const int &data)
89 {
90     TListNode *newPtr = new TListNode();
91
92     newPtr->Data = data;
93     if (isEmpty())
94         firstPtr = lastPtr = newPtr;
95     else {
96         lastPtr->nextPtr = newPtr;
97         lastPtr = newPtr;
98     }
99 }
100 //-----
101 int TList::removeFromFront(int &data)
102 {
103     if (isEmpty())
104         return 0;
105     else {
106         TListNode *tempPtr = firstPtr;
107
108         if (firstPtr == lastPtr)
109             firstPtr = lastPtr = NULL;
110         else
111             firstPtr = firstPtr->nextPtr;
112
113         data = tempPtr->Data;
114         delete tempPtr;
115         return 1;
116     }
117 }
118 //-----
119 int TList::removeFromBack(int &data)
120 {
121     if (isEmpty())
122         return 0;
123     else {
124         TListNode *tempPtr = lastPtr;
125
126         if (firstPtr == lastPtr)
127             firstPtr = lastPtr = NULL;
128         else {
129             TListNode *currentPtr = firstPtr;
130
131             while (currentPtr->nextPtr != lastPtr)
132                 currentPtr = currentPtr->nextPtr;
133
134             lastPtr = currentPtr;
135             currentPtr->nextPtr = NULL;
136         }
137         data = tempPtr->Data;
138         delete tempPtr;
139         return 1;
140     }
141 }
142 //-----
143 void TList::sortElements()
144 {
145     TListNode *firstTempPtr, *secondTempPtr, *thirdTempPtr, *secondPtr;
146     bool flaw;

```

```

147
148     if (firstPtr != NULL)
149         do {
150             flaw = false;
151
152             secondPtr = firstPtr->nextPtr;
153             if (secondPtr != NULL && firstPtr->Data > secondPtr->Data) {
154                 firstPtr->nextPtr = secondPtr->nextPtr;
155                 secondPtr->nextPtr = firstPtr;
156                 firstPtr = secondPtr;
157                 flaw = true;
158             }
159
160             firstTempPtr = firstPtr;
161             secondTempPtr = firstTempPtr->nextPtr;
162             thirdTempPtr = secondTempPtr->nextPtr;
163             while (thirdTempPtr != NULL) {
164                 if (secondTempPtr->Data > thirdTempPtr->Data) {
165                     firstTempPtr->nextPtr = thirdTempPtr;
166                     secondTempPtr->nextPtr = thirdTempPtr->nextPtr;
167                     thirdTempPtr->nextPtr = secondTempPtr;
168                     flaw = true;
169                 }
170                 firstTempPtr = firstTempPtr->nextPtr;
171                 secondTempPtr = firstTempPtr->nextPtr;
172                 thirdTempPtr = secondTempPtr->nextPtr;
173             }
174             lastPtr = secondTempPtr;
175         } while (flaw);
176     }
177     //-----
178     void TList::print() const
179     {
180         TListNode *currentPtr = firstPtr;
181
182         if (isEmpty()) {
183             cout << "The list is empty!" << endl << endl;
184             return;
185         }
186
187         cout << "List = ";
188         while (currentPtr != NULL) {
189             cout << currentPtr->Data << ' ';
190             currentPtr = currentPtr->nextPtr;
191         }
192         cout << endl << endl;
193     }
194     //-----
195

```