

```
1: program LinkedList;
2:
3: uses
4:   sysUtils;
5: type
6:   NodePtr = ^Node;
7:
8:   Node = record
9:     data : Integer;
10:    next : NodePtr;
11:  end;
12:
13:  LinkedList = record
14:    start : NodePtr;
15:    finish : NodePtr;
16:  end;
17:
18:  // Add enumerated type to change the operator in the find Node function
19:
20: procedure CreateList(var list: LinkedList);
21: begin
22:   list.start := nil;
23:   list.finish := nil;
24: end;
25:
26:
27: // Create Node
28: function CreateNode(data : Integer; next : NodePtr): NodePtr;
29: begin
30:   New(result);
31:   result^.data := data;
32:   result^.next := next;
33: end;
34:
35: function NodeCount(list: NodePtr; value: Integer): Integer;
36: begin
37:   result := 0;
38:   while (Assigned(list)) AND (list^.data <> value) do
39:     begin
40:       list := list^.next;
41:       result := result + 1;
42:     end;
43: end;
44:
45:
46: function FindNode(list: NodePtr; value: Integer): NodePtr;
47: var
48:   current : NodePtr;
49: begin
50:   result := nil;
51:   current := list;
52:
53:   while (Assigned(current)) AND (current^.data <> value) do
54:     begin
55:       current := current^.next;
56:       result := current;
57:     end;
58: end;
59:
60:
61: // new find previous
62: // function FindPreviousNode(list, ofNode: NodePtr; val: Integer) : NodePtr;
63: // var
```

```
64: // current, previous : NodePtr;
65: //
66: // begin
67: //   current := list;
68: //   previous := nil;
69: //   WriteLn('val in FindPrevNode', val);
70: //   while (current <> ofNode) and (current <> nil) do
71: //     begin
72: //       // pass in the value and it will return the previous node
73: //       previous := current;
74: //       current := current^.next;
75: //     end;
76: //   result := previous;
77: // end;
78:
79:
80:
81: function FindPreviousNode(list: NodePtr; val: Integer) : NodePtr;
82: var
83:   i : Integer;
84: begin
85:   WriteLn('val in FindPrevNode', val);
86:   i := 0;
87:   for i:= 0 to val-2 do
88:     begin
89:       // pass in the value and it will return the previous node
90:       list := list^.next;
91:       result := list;
92:     end;
93: end;
94:
95: function FindNextNode(list: NodePtr; val: Integer) : NodePtr;
96: var
97:   i : Integer;
98: begin
99:   WriteLn('val in FindPrevNode', val);
100:  i := 0;
101:  for i:= 0 to val-1 do
102:    begin
103:      // pass in the value and it will return the previous node
104:      list := list^.next;
105:      result := list;
106:    end;
107: end;
108:
109: // Dispose all nodes
110: procedure DisposeNodes(var start: NodePtr);
111: begin
112:   if start <> nil then
113:     begin
114:       DisposeNodes(start^.next);
115:       Dispose(start);
116:       start := nil;
117:     end;
118: end;
119:
120: // Insert before procedure determined by parameters
121: procedure InsertBefore(list : NodePtr; beforeVal, val : Integer);
122: var
123:   nodeBefore, temp: NodePtr;
124: begin
125:   nodeBefore := FindPreviousNode(list, NodeCount(list, beforeVal));
126:   temp := CreateNode(val, nodeBefore^.next);
```

```
127:   nodeBefore^.next := temp;
128: end;
129:
130: // Insert after procedure determined by parameters
131: procedure InsertAfter(list : NodePtr; beforeVal, val : Integer);
132: var
133:   nodeBefore, temp: NodePtr;
134: begin
135:   // find the node before
136:   nodeBefore := FindNextNode(list, NodeCount(list, beforeVal));
137:
138:   temp := CreateNode(val, nodeBefore^.next);
139:   nodeBefore^.next := temp;
140: end;
141:
142: // Insert At Start of List
143: procedure PrependNode(var linked: LinkedList; value: Integer);
144: var
145:   temp : NodePtr;
146: begin
147:   temp := CreateNode(value, linked.start);
148:   linked.start := temp;
149:   if linked.finish = nil then
150:   begin
151:     linked.finish := temp;
152:   end;
153: end;
154:
155: procedure AppendNode(var linked: LinkedList; value: Integer);
156: var
157:   temp : NodePtr;
158: begin
159:   temp := CreateNode(value, nil);
160:   if linked.finish <> nil then
161:   begin
162:     linked.finish^.next := temp;
163:   end
164:   else
165:   begin
166:     // list is empty
167:     linked.start := temp;
168:   end;
169:
170:   linked.finish := temp;
171: end;
172:
173: // Print Nodes
174: procedure PrintBackTo(n: NodePtr);
175: begin
176:   if (n <> nil) then
177:   begin
178:     PrintBackTo(n^.next);
179:     Write(' <- ', n^.data);
180:   end
181:   else
182:   begin
183:     Write('nil');
184:   end;
185: end;
186:
187: // Print From Node
188: // In: NodePtr = list.start
189: procedure PrintFrom(n: NodePtr);
```

```
190: begin
191:   if n <> nil then
192:   begin
193:     Write(n^.data, ' -> ');
194:     PrintFrom(n^.next);
195:   end
196:   else
197:   begin
198:     WriteLn('nil');
199:   end;
200:   //WriteLn();
201: end;
202:
203: // Count nodes
204: function Count(n: NodePtr):Integer;
205: var
206:   current: NodePtr;
207: begin
208:   current := n;
209:   result := 0;
210:   while (current <> nil) do
211:   begin
212:     result +=1;
213:     current := current^.next;
214:   end;
215: end;
216:
217:
218: procedure Main();
219: var
220:   list : LinkedList;
221:   find : NodePtr;
222: begin
223:   CreateList(list);
224:   PrependNode(list, 1);
225:   PrependNode(list, 5);
226:   PrependNode(list, 10);
227:   PrependNode(list, 15);
228:   PrependNode(list, 20);
229:   AppendNode(list, 99);
230:   WriteLn('=====');
231:   PrintFrom(list.start);
232:   WriteLn('=====');
233:   WriteLn('Start of the list data ', list.start^.data);
234:   WriteLn('Finish of the list data ', list.finish^.data);
235:   PrependNode(list, 30);
236:   WriteLn('Start of the list data ', list.start^.data);
237:   PrintFrom(list.start);
238:
239:   find := FindNode(list.start, 10);
240:
241:   FindPreviousNode(list.start, 2);
242:   WriteLn('Node Count : ', NodeCount(list.start, 15));
243:
244:
245:   WriteLn('=====');
246:   WriteLn('find ', find^.data);
247:
248:   PrintFrom(list.start);
249:   WriteLn('=====');
250:
251:   find := FindPreviousNode(list.start, NodeCount(list.start, 10));
252:   WriteLn('Find Previous Node Function ', find^.data);
```

LinkedList.pas

```
253:    //WriteLn('Find Node ', FindNode(list,20));
254:    WriteLn('=====');
255:
256:    InsertBefore(list.start, 15, 18);
257:    InsertAfter(list.start, 15, 22);
258:    PrintFrom(list.start);
259:
260:
261: end;
262:
263: // Main executable
264: begin
265:     Main();
266: end.
```