```
 1: unit LstDEnc;
 2:
 3: interface
 4:
 5: type
 6:    { Tipo de chave do item da lista }
 7:    Tipo_Chave = integer;
 8:
 9:    { Tipo do item }
10:    Tipo_Item = record
11:                   Chave   : Tipo_Chave;
12:                   Dado    : String[30];
13:                end;
14:
15:    { Tipo do apontador para um item da lista }
16:    Pont_Item_Lista = ^Item_Lista;
17:
18:    { Tipo do item da lista }
19:    Item_Lista = record
20:                   Item : Tipo_Item;
21:                   Proximo,
22:                   Anterior : Pont_Item_Lista
23:                 end;
24:
25:    { Tipo da lista Duplamente Encadeada }
26:    Lista_Duplamente_Enc = record
27:                              Cabeca : Pont_Item_Lista;
28:                              Cauda  : Pont_Item_Lista;
29:                              Tamanho : integer
30:                            end;
31:
32: procedure Inicializar (var Lista : Lista_Duplamente_Enc);
33: function Inserir (Item : Tipo_Item; var Lista : Lista_Duplamente_Enc) :
    boolean;
34: function Remover (Chave : Tipo_Chave; var Lista : Lista_Duplamente_Enc) :
    boolean;
35: function Alterar (Item : Tipo_Item; var Lista : Lista_Duplamente_Enc) :
    boolean;
36: procedure Obter (Chave : Tipo_Chave; var Lista : Lista_Duplamente_Enc;
37:                  var Item : Tipo_Item; var Sucesso : boolean);
38: procedure Apagar (var Lista : Lista_Duplamente_Enc);
39: function Tamanho (var Lista : Lista_Duplamente_Enc) : integer;
40: function Cheia (var Lista : Lista_Duplamente_Enc) : boolean;
41: function Vazia (var Lista : Lista_Duplamente_Enc) : boolean;
42:
43: implementation
44:
45: procedure Inicializar (var Lista : Lista_Duplamente_Enc);
46: {
47:   Objetivo: Inicializa a lista passada, fazendo com que a cabeca e a cauda
48:             da lista apontem para nil.
49: }
50: begin
51:    Lista.Cabeca := nil;
52:    Lista.Cauda := nil;
53:    Lista.Tamanho := 0
54: end;
55:
56:
57: function Inserir (Item : Tipo_Item; var Lista : Lista_Duplamente_Enc) :
    boolean;
```

```
 58: {
 59:   Objetivo: Insere o item passado como parametro na lista passada.
 60:             Se a lista ja estiver cheia, a funcao Inserir retorna false.
 61: }
 62: var
 63:    PNovo : Pont_Item_Lista;
 64: begin
 65:    if Cheia(Lista) then
 66:       Inserir := false
 67:    else
 68:       begin
 69:          New(PNovo);
 70:          PNovo^.Item := Item;
 71:          PNovo^.Proximo := Lista.Cabeca;
 72:          PNovo^.Anterior := nil;
 73:
 74:          if Lista.Cabeca <> nil then
 75:             Lista.Cabeca^.Anterior := PNovo;
 76:
 77:          Lista.Cabeca := PNovo;
 78:
 79:          if Lista.Cauda = nil then
 80:             Lista.Cauda := PNovo;
 81:
 82:          inc(Lista.Tamanho);
 83:          Inserir := true
 84:       end
 85: end;
 86:
 87:
 88: function Remover (Chave : Tipo_Chave; var Lista : Lista_Duplamente_Enc) :
     boolean;
 89: {
 90:   Objetivo: Remove o item cuja chave coincide com o parametro Chave
 91:             passado. Caso nao haja um item com essa chave, retorna
 92:             false. Se o item foi removido, retorna true.
 93: }
 94: var
 95:    PAtual : Pont_Item_Lista;
 96: begin
 97:    Remover := false;
 98:
 99:    PAtual := Lista.Cabeca;
100:
101:    while PAtual <> nil do
102:    begin
103:       if PAtual^.Item.Chave = Chave then
104:          begin
105:             if PAtual = Lista.Cabeca then
106:                Lista.Cabeca := PAtual^.Proximo;
107:
108:             if PAtual = Lista.Cauda then
109:                Lista.Cauda := PAtual^.Anterior;
110:
111:             if PAtual^.Proximo <> nil then
112:                PAtual^.Proximo^.Anterior := PAtual^.Anterior;
113:
114:             if PAtual^.Anterior <> nil then
115:                PAtual^.Anterior^.Proximo := PAtual^.Proximo;
116:
117:             Dispose(PAtual);
```

```
118:             dec(Lista.Tamanho);
119:             Remover := true;
120:             break
121:          end
122:       else
123:          PAtual := PAtual^.Proximo
124:    end;
125: end;
126:
127:
128: function ObterNo (Chave : Tipo_Chave; var Lista : Lista_Duplamente_Enc) :
     Pont_Item_Lista;
129: {
130:   Objetivo: Retorna um apontador para o No que contem o Item com a chave
131:             igual a passada. Caso nao seja encontrado, a funcao retorna nil
132: }
133: var
134:    PAtual : Pont_Item_Lista;
135: begin
136:    ObterNo := nil;
137:    PAtual := Lista.Cabeca;
138:
139:    while PAtual <> nil do
140:       if PAtual^.Item.Chave = Chave then
141:          begin
142:             ObterNo := PAtual;
143:             break
144:          end
145:       else
146:          PAtual := PAtual^.Proximo
147: end;
148:
149:
150: procedure Obter (Chave : Tipo_Chave; var Lista : Lista_Duplamente_Enc;
151:                  var Item : Tipo_Item; var Sucesso : boolean);
152: {
153:   Objetivo: Procura na lista usando a chave passada. Caso encontre
154:             Sucesso contem o valor true e Item contem o Item obtido.
155:             Caso contrario, Sucesso retorna true e Item nao e alterado
156: }
157: var
158:    PAtual : Pont_Item_Lista;
159: begin
160:    Sucesso := false;
161:    PAtual := ObterNo(Chave, Lista);
162:
163:    if PAtual <> nil then
164:       begin
165:          Item := PAtual^.Item;
166:          Sucesso := true
167:       end
168: end;
169:
170:
171: function Alterar (Item : Tipo_Item; var Lista : Lista_Duplamente_Enc) :
     boolean;
172: {
173:   Objetivo: Altera os dados de um item existente na lista passada
174:             de forma que fique igual ao do item passado como parametro.
175:             Se o item for encontrado e alterado, retorna true. Caso
176:             contrario, retorna false.
```

```
177: }
178: var
179:    PAtual : Pont_Item_Lista;
180: begin
181:    Alterar := false;
182:    PAtual := ObterNo(Item.Chave, Lista);
183:
184:    if PAtual <> nil then
185:       begin
186:          PAtual^.Item := Item;
187:          Alterar := true
188:       end
189: end;
190:
191:
192: procedure Apagar (var Lista : Lista_Duplamente_Enc);
193: {
194:   Objetivo: Apaga a lista passada
195: }
196: var
197:    PAtual : Pont_Item_Lista;
198: begin
199:    if Vazia(Lista) then
200:       exit;
201:
202:    PAtual := Lista.Cabeca;
203:
204:    while PAtual <> Lista.Cauda do
205:    begin
206:       PAtual := PAtual^.Proximo;
207:       Dispose(PAtual^.Anterior)
208:    end;
209:
210:    Dispose(Lista.Cauda);
211:
212:    Lista.Cabeca := nil;
213:    Lista.Cauda := nil;
214:    Lista.Tamanho := 0
215: end;
216:
217:
218: function Tamanho (var Lista : Lista_Duplamente_Enc) : integer;
219: {
220:   Objetivo: Retorna o tamanho da lista passada
221: }
222: begin
223:    Tamanho := Lista.Tamanho
224: end;
225:
226:
227: function Cheia (var Lista : Lista_Duplamente_Enc) : boolean;
228: {
229:   Objetivo: Retorna true se nao ha mais memoria disponivel
230:             para inserir um item na lista
231: }
232: begin
233:    Cheia := MaxAvail < SizeOf(Item_Lista)
234: end;
235:
236:
237: function Vazia (var Lista : Lista_Duplamente_Enc) : boolean;
```

```
238: {
239:   Objetivo: Retorna true se a lista esta vazia
240: }
241: begin
242:    Vazia := Lista.Tamanho = 0
243: end;
244:
245: end.
```