```
 1: unit LstEnc;
 2:
 3: interface
 4:
 5: type
 6:    { Tipo de chave do item da lista }
 7:    Tipo_Chave = longint;
 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 : Pont_Item_Lista
22:                 end;
23:
24:    { Tipo da lista Encadeada }
25:    Lista_Encadeada = record
26:                         Cabeca : Pont_Item_Lista;
27:                       end;
28:
29: procedure Inicializar (var Lista : Lista_Encadeada);
30: function Inserir (Item : Tipo_Item; var Lista : Lista_Encadeada) : boolean;
31: function Remover (Chave : Tipo_Chave; var Lista : Lista_Encadeada) : boolean;
32: function Alterar (Item : Tipo_Item; var Lista : Lista_Encadeada) : boolean;
33: procedure Obter (Chave : Tipo_Chave; var Lista : Lista_Encadeada;
34:                  var Item : Tipo_Item; var Sucesso : boolean);
35: procedure Apagar (var Lista : Lista_Encadeada);
36: function Tamanho (var Lista : Lista_Encadeada) : longint;
37: function Vazia (var Lista : Lista_Encadeada) : boolean;
38:
39: implementation
40:
41: procedure Inicializar (var Lista : Lista_Encadeada);
42: {
43:   Objetivo: Inicializa a lista passada, fazendo com que a cabeca da lista
44:             aponte para nil.
45: }
46: begin
47:    ReturnNilIfGrowHeapFails := true;
48:    Lista.Cabeca := nil
49: end;
50:
51:
52: function AlocarItem (var PItem : Pont_Item_Lista) : boolean;
53: {
54:   Objetivo: Tentar alocar um Item usando o apontador passado,
55:             retornando true se conseguir e false caso contrario.
56: }
57: begin
58:    New(PItem);
59:    AlocarItem := PItem <> nil;
60: end;
61:
```

```
62:
63: function Inserir (Item : Tipo_Item; var Lista : Lista_Encadeada) : boolean;
64: {
65:   Objetivo: Insere o item passado como parametro na lista passada.
66:             Se a lista ja estiver cheia, a funcao Inserir retorna false.
67: }
68: var
69:    PNovo : Pont_Item_Lista;
70: begin
71:    if AlocarItem(PNovo) then
72:       begin
73:          PNovo^.Item := Item;
74:          PNovo^.Proximo := Lista.Cabeca;
75:          Lista.Cabeca := PNovo;
76:          Inserir := true
77:       end
78:    else
79:       Inserir := false
80: end;
81:
82:
83: function Remover (Chave : Tipo_Chave; var Lista : Lista_Encadeada) : boolean;
84: {
85:   Objetivo: Remove o item cuja chave coincide com o parametro Chave
86:             passado. Caso nao haja um item com essa chave, retorna
87:             false. Se o item foi removido, retorna true.
88: }
89: var
90:    PAtual, PAnterior : Pont_Item_Lista;
91: begin
92:    Remover := false;
93:
94:    if Vazia(Lista) then
95:       exit;
96:
97:    PAnterior := Lista.Cabeca;
98:
99:    if Lista.Cabeca^.Item.Chave = Chave then
100:      begin
101:         { Remove o item da cabeca da lista e a cabeca passa a ser
102:           o item seguinte ao removido}
103:         Lista.Cabeca := Lista.Cabeca^.Proximo;
104:         dispose(PAnterior);
105:         Remover := true
106:      end
107:   else
108:      { Percorre a lista ate encontrar um item com a chave procurada.
109:        Remove o item e corrige o apontador do item anterior para
110:        apontar para o proximo item }
111:      while PAnterior^.Proximo <> nil do
112:      begin
113:         PAtual := PAnterior^.Proximo;
114:
115:         if PAtual^.Item.Chave = Chave then
116:            begin
117:               PAnterior^.Proximo := PAtual^.Proximo;
118:               Dispose(PAtual);
119:               Remover := true;
120:               break
121:            end;
122:
```

```
123:          PAnterior := PAtual;
124:       end
125: end;
126:
127:
128: function Alterar (Item : Tipo_Item; var Lista : Lista_Encadeada) : boolean;
129: {
130:   Objetivo: Altera os dados de um item existente na lista passada
131:             de forma que fique igual ao do item passado como parametro.
132:             Se o item for encontrado e alterado, retorna true. Caso
133:             contrario, retorna false.
134: }
135: var
136:    PAtual : Pont_Item_Lista;
137: begin
138:    Alterar := false;
139:    PAtual := Lista.Cabeca;
140:
141:    while PAtual <> nil do
142:       if PAtual^.Item.Chave = Item.Chave then
143:          begin
144:             PAtual^.Item := Item;
145:             Alterar := true;
146:             break
147:          end
148:       else
149:          PAtual := PAtual^.Proximo
150: end;
151:
152:
153: procedure Obter (Chave : Tipo_Chave; var Lista : Lista_Encadeada;
154:                  var Item : Tipo_Item; var Sucesso : boolean);
155: {
156:   Objetivo: Procura na lista usando a chave passada. Caso encontre
157:             Sucesso contem o valor true e Item contem o Item obtido.
158:             Caso contrario, Sucesso retorna true e Item nao e alterado
159: }
160: var
161:    PAtual : Pont_Item_Lista;
162: begin
163:    Sucesso := false;
164:    PAtual := Lista.Cabeca;
165:
166:    while PAtual <> nil do
167:       if PAtual^.Item.Chave = Chave then
168:          begin
169:             Item := PAtual^.Item;
170:             Sucesso := true;
171:             break
172:          end
173:       else
174:          PAtual := PAtual^.Proximo
175: end;
176:
177:
178: procedure Apagar (var Lista : Lista_Encadeada);
179: {
180:   Objetivo: Apaga a lista passada
181: }
182: var
183:    PAtual, PApagar : Pont_Item_Lista;
```

```
184: begin
185:    PAtual := Lista.Cabeca;
186:
187:    while PAtual <> nil do
188:    begin
189:       PApagar := PAtual;
190:       PAtual := PAtual^.Proximo;
191:       Dispose(PApagar)
192:    end;
193:
194:    Lista.Cabeca := nil
195: end;
196:
197:
198: function Tamanho (var Lista : Lista_Encadeada) : longint;
199: {
200:   Objetivo: Retorna o tamanho da lista passada
201: }
202: var
203:    P : Pont_Item_Lista;
204:    Cont : longint;
205: begin
206:    P := Lista.Cabeca;
207:    Cont := 0;
208:
209:    while P <> nil do
210:    begin
211:       inc(Cont);
212:       P := P^.Proximo
213:    end;
214:
215:    Tamanho := Cont;
216: end;
217:
218:
219: function Vazia (var Lista : Lista_Encadeada) : boolean;
220: {
221:   Objetivo: Retorna true se a lista esta vazia (Cabeca = nil)
222: }
223: begin
224:    Vazia := Lista.Cabeca = nil
225: end;
226:
227: end.
```