```
 1: unit TabHash;
 2: interface
 3: const
 4:    MAX = 100;
 5: type
 6:    Tipo_da_Chave = longint;
 7:    Tipo_do_Dado = record
 8:                      Chave : Tipo_da_Chave;
 9:                      { Outras Informacoes }
10:                   end;
11:    TSituacao = (NuncaUsada, Desocupada, Ocupada);
12:    Item_Tab_Hash = record
13:                      Situacao : TSituacao;
14:                      Dado : Tipo_do_Dado;
15:                    end;
16:    Pos_Tab_Hash = 1..MAX;
17:    Tab_Hash = array[Pos_Tab_Hash] of Item_Tab_Hash;
18:
19:
20:    procedure Inicializar(var T : Tab_Hash);
21:    function Hash(Chave : Tipo_da_Chave) : Pos_Tab_Hash;
22:    function Tamanho(var T : Tab_Hash) : longint;
23:    function Cheia(var T : Tab_Hash) : boolean;
24:    function Vazia(var T : Tab_Hash) : boolean;
25:    function Inserir(var T : Tab_Hash; Dado : Tipo_do_Dado) : boolean;
26:    function Remover(var T : Tab_Hash; Chave : Tipo_da_Chave) : boolean;
27:    function Obter(var T : Tab_Hash; Chave : Tipo_da_Chave;
28:                   var Dado : Tipo_do_Dado) : boolean;
29:    function Alterar(var T : Tab_Hash; Dado : Tipo_do_Dado) : boolean;
30:
31: implementation
32:
33: (*************************************************************************
    )
34:
35: procedure Inicializar(var T : Tab_Hash);
36: {
37:   Objetivo: Inicializa a tabela hash passada, fazendo com que a
38:             situacao de todas as posicoes da tabela sejam marcadas
39:             como nunca usadas (NuncaUsada)
40: }
41: var
42:    I : Pos_Tab_Hash;
43: begin
44:    for I := 1 to MAX do
45:       T[I].Situacao := NuncaUsada;
46: end;
47:
48: (*************************************************************************
    )
49:
50: function Hash(Chave : Tipo_da_Chave) : Pos_Tab_Hash;
51: {
52:   Objetivo: Retorna a posição da tabela correspondente ao valor de chave
53:             passado. Usa a funcao: (Chave mod MAX) + 1
54: }
55: begin
56:    Hash := (Chave mod MAX) + 1;
57: end;
58:
59: (*************************************************************************
    )
```

```
60:
61: function Tamanho(var T : Tab_Hash) : longint;
62: {
63:   Objetivo: Retorna o numero de posicoes da tabela ocupadas, ou seja,
64:             o numero de elementos que possuem a situacao Ocupada
65: }
66: var
67:    I : Pos_Tab_Hash;
68:    Cont : longint;
69: begin
70:    Cont := 0;
71:    for I := 1 to MAX do
72:       if T[I].Situacao = Ocupada then
73:          inc(Cont);
74:
75:    Tamanho := Cont;
76: end;
77:
78: (*************************************************************************
    )
79:
80: function Cheia(var T : Tab_Hash) : boolean;
81: {
82:   Objetivo: Retorna true se todas as posicoes da tabela estiverem
83:             ocupadas (Ocupada)
84: }
85: begin
86:    Cheia := Tamanho(T) = MAX
87: end;
88:
89: (*************************************************************************
    )
90:
91: function Vazia(var T : Tab_Hash) : boolean;
92: {
93:   Objetivo: Retorna todas as posicoes da tabela estiverem disponiveis
94:             (Desocupada ou NuncaUsada)
95: }
96: begin
97:    Vazia := Tamanho(T) = 0
98: end;
99:
100: (*************************************************************************
    )
101:
102: function ProximaPosicao(I : Pos_Tab_Hash) : Pos_Tab_Hash;
103: {
104:   Objetivo: Retorna a proxima posicao a ser analisada na presenca de
105:             colisoes, baseada na posicao passada
106: }
107: begin
108:    ProximaPosicao := (I mod MAX) + 1;
109: end;
110:
111: (*************************************************************************
    )
112:
113: function Inserir(var T : Tab_Hash; Dado : Tipo_do_Dado) : boolean;
114: {
115:   Objetivo: Insere na tabela T, o dado contido no parametro Dado. Se ja
    existir
```

```
116:             alguma chave igual na tabela, retorna false para indicar que nao
117:             foi possivel inserir. Caso contrario, coloca o dado na primeira
118:             posicao disponivel (Desocupada ou NuncaUsada) e retorna true
119: }
120: var
121:    I, Pos_Insercao : Pos_Tab_Hash;
122:    Cont : longint;
123: begin
124:    Inserir := false;
125:    I := Hash(Dado.Chave);
126:    Cont := 0;
127:
128:    { Procura a primeira posicao livre (Desocupada ou NuncaUsada) disponivel.
129:      Se encontrar alguma chave igual sai da funcao retornando false }
130:    while (T[I].Situacao = Ocupada) and (Cont < MAX) do
131:       if T[I].Dado.Chave = Dado.Chave then
132:          exit
133:       else
134:          begin
135:             I := ProximaPosicao(I);
136:             inc(Cont);
137:          end;
138:
139:    { Verifica se a tabela esta cheia (todas as posicoes ocupadas) }
140:    if Cont = MAX then
141:       exit;
142:
143:    Pos_Insercao := I;
144:
145:    { Procura por uma chave ja existente ate encontrar uma posicao da tabela
146:      que nunca tenha sido usada ou tenha percorrido toda a tabela. Isso
147:      garante que nao ha uma chave igual na tabela }
148:    while (T[I].Situacao in [Desocupada, Ocupada]) and (Cont < MAX) do
149:    begin
150:       { Se o elemento esta ocupado e a chave coincide, o dado nao deve
151:         ser inserido }
152:       if (T[I].Situacao = Ocupada) and (T[I].Dado.Chave = Dado.Chave) then
153:             exit;
154:
155:        I := ProximaPosicao(I);
156:        inc(Cont);
157:    end;
158:
159:    { Coloca o dado na primeira posição vaga encontrada }
160:    T[Pos_Insercao].Dado := Dado;
161:    T[Pos_Insercao].Situacao := Ocupada;
162:    Inserir := true;
163: end;
164:
165:
166: (*************************************************************************
     )
167:
168: function EncontrarPos(var T : Tab_Hash; Chave : Tipo_da_Chave;
169:                       var Pos : Pos_Tab_Hash) : boolean;
170: {
171:   Objetivo: Procura pela chave passada. Se existir, o parametro
172:             Pos contem a posicao e retorna true. Caso contrario,
173:             apenas retorna false
174: }
175: var
```

```
176:    I : Pos_Tab_Hash;
177:    Cont : longint;
178: begin
179:    EncontrarPos := false;
180:    I := Hash(Chave);
181:    Cont := 0;
182:
183:    { Comeca a procurar na posicao retornada pela funcao Hash e avanca
184:      ate encontrar uma posicao que nunca foi usada (isso indica que
185:      dali para frente, nao ha possibilidade de encontrar a chave
186:      procurada) ou ate percorrer toda a tabela }
187:    while (T[I].Situacao <> NuncaUsada) and (Cont < MAX) do
188:    begin
189:       if (T[I].Situacao = Ocupada) and (T[I].Dado.Chave = Chave) then
190:          begin
191:             EncontrarPos := true;
192:             Pos := I;
193:             exit
194:          end;
195:
196:       inc(Cont);
197:       I := ProximaPosicao(I);
198:    end;
199: end;
200:
201: (*************************************************************************
     )
202:
203: function Remover(var T : Tab_Hash; Chave : Tipo_da_Chave) : boolean;
204: {
205:   Objetivo: Remove o dado contido na tabela que possui a chave passada.
206:             Se existir, retorna true e a posicao do array e marcada como
207:             Desocupada. Caso contrario, apenas retorna false
208: }
209: var
210:    Pos : Pos_Tab_Hash;
211: begin
212:    if EncontrarPos(T, Chave, Pos) then
213:       begin
214:          Remover := true;
215:          T[Pos].Situacao := Desocupada
216:       end
217:    else
218:       Remover := false;
219: end;
220:
221: (*************************************************************************
     )
222:
223: function Obter(var T : Tab_Hash; Chave : Tipo_da_Chave;
224:                var Dado : Tipo_do_Dado) : boolean;
225: {
226:   Objetivo: Se a tabela contem algum elemento com a chave especificada,
227:             coloca o dado correspondente no parametro Dado e retorna true.
228:             Caso contrario, apenas retorna false
229: }
230: var
231:    Pos : Pos_Tab_Hash;
232: begin
233:    if EncontrarPos(T, Chave, Pos) then
234:       begin
```

```
235:          Obter := true;
236:          Dado := T[Pos].Dado
237:       end
238:    else
239:       Obter := false;
240: end;
241:
242: (*************************************************************************
     )
243:
244: function Alterar(var T : Tab_Hash; Dado : Tipo_do_Dado) : boolean;
245: {
246:   Objetivo: Se a tabela contem algum elemento com a chave especificada,
247:             coloca o dado contido no parametro Dado na posicao correta
248:             da tabela e retorna true. Caso contrario, apenas retorna false
249: }
250: var
251:    Pos : Pos_Tab_Hash;
252: begin
253:    if EncontrarPos(T, Dado.Chave, Pos) then
254:       begin
255:          Alterar := true;
256:          T[Pos].Dado := Dado
257:       end
258:    else
259:       Alterar := false;
260: end;
261:
262: end.
```