```
 1: unit PilhaE;
 2: interface
 3:
 4: type
 5:    Tipo_do_Dado = record
 6:                      Dado : Char;
 7:                   end;
 8: type
 9:    PNo = ^No;
10:    No  = record
11:             Info : Tipo_do_Dado;
12:             Prox : PNo;
13:          end;
14: type
15:    Pilha_Enc = record
16:                   Topo    : PNo;
17:                   Tamanho : longint;
18:                end;
19:
20: procedure Inicializar (var Pilha : Pilha_Enc);
21: function Vazia (var Pilha : Pilha_Enc) : boolean;
22: function Tamanho (var Pilha : Pilha_Enc) : longint;
23: procedure Empilhar (Info : Tipo_do_Dado; var Pilha : Pilha_Enc);
24: procedure Desempilhar (var Info : Tipo_do_Dado; var Pilha : Pilha_Enc);
25: procedure Topo (var Info : Tipo_do_Dado; var Pilha : Pilha_Enc);
26:
27: implementation
28:
29: procedure Inicializar (var Pilha : Pilha_Enc);
30: {
31:   Objetivos: Inicializa a Pilha, atribuindo nil ao topo e zerando o
32:              tamanho da Pilha
33: }
34: begin
35:    Pilha.Topo := nil;
36:    Pilha.Tamanho := 0;
37:    ReturnNilIfGrowHeapFails := true
38: end;
39:
40:
41: function AlocarItem (var PItem : PNo) : boolean;
42: {
43:   Objetivo: Tentar alocar um Item usando o apontador passado,
44:             retornando true se conseguir e false caso contrario.
45: }
46: begin
47:    New(PItem);
48:    AlocarItem := PItem <> nil;
49: end;
50:
51:
52: function Vazia (var Pilha : Pilha_Enc) : boolean;
53: {
54:   Objetivos: Retorna true se o numero de Nos na Pilha for igual a zero. Caso
55:              contrario, retorna false
56: }
57: begin
58:    Vazia := Pilha.Tamanho = 0;
59: end;
60:
61:
```

```
 62: function Tamanho (var Pilha : Pilha_Enc) : longint;
 63: {
 64:  Objetivos: Retorna o numero de Nos existentes na Pilha.
 65: }
 66: begin
 67:    Tamanho := Pilha.Tamanho;
 68: end;
 69:
 70:
 71: procedure Empilhar (Info : Tipo_do_Dado; var Pilha : Pilha_Enc);
 72: {
 73:   Objetivos: Empilha um No no topo da pilha.
 74: }
 75: var
 76:    PTemp : PNo;
 77: begin
 78:    { Tenta alocar um item na Pilha. Se estiver cheia, da uma mensagem
 79:      e sai do programa }
 80:    if not AlocarItem(PTemp) then
 81:       begin
 82:          writeln('ERRO: Overflow da pilha.');
 83:          halt;
 84:       end;
 85:
 86:    { Coloca a informacoes no no alocado }
 87:    PTemp^.Info := Info;
 88:
 89:    { Faz com que o novo No seja o Topo da Pilha }
 90:    PTemp^.Prox := Pilha.Topo;
 91:    Pilha.Topo := PTemp;
 92:
 93:    { Contador de Nos aumenta de um. }
 94:    inc(Pilha.Tamanho);
 95: end;
 96:
 97:
 98: procedure Desempilhar (var Info : Tipo_do_Dado; var Pilha : Pilha_Enc);
 99: {
100:   Objetivos: Retira o No do Topo da Pilha
101: }
102: var
103:    PTemp : PNo;
104: begin
105:     { Testa se a Pilha esta vazia. Se estiver da uma mensagem e sai do
    programa }
106:    if Vazia(Pilha) then
107:       begin
108:          writeln('ERRO: Underflow da pilha.');
109:          halt;
110:       end;
111:
112:    { Coloca em Info o conteudo do No no topo da Pilha. O No que esta abaixo
113:      do Topo passa a ser o Topo da Pilha e o primeiro No, desalocado da
    memoria.
114:      Depois, o contador de Nos diminui de um. }
115:    Info := Pilha.Topo^.Info;
116:    PTemp := Pilha.Topo;
117:    Pilha.Topo := Pilha.Topo^.Prox;
118:    Dispose(PTemp);
119:    dec(Pilha.Tamanho);
120: end;
```

```
121:
122:
123: procedure Topo (var Info : Tipo_do_Dado; var Pilha : Pilha_Enc);
124: {
125:   Objetivo: Retorna o No que se encontra no Topo da Pilha sem, no entanto,
126:             retira-lo da Pilha
127: }
128: begin
129:     { Testa se a Pilha esta vazia. Se estiver da uma mensagem e sai do
     programa }
130:    if Vazia(Pilha) then
131:       begin
132:          writeln('ERRO: Topo da Pilha inexistente');
133:          halt;
134:       end;
135:
136:    { Coloca em Info o conteudo do No no topo da Pilha }
137:    Info := Pilha.Topo^.Info;
138: end;
139:
140: end.
```