```
 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 : integer;
18:                end;
19:
20: procedure Inicializar (var Pilha : Pilha_Enc);
21: function Vazia (var Pilha : Pilha_Enc) : boolean;
22: function Cheia (var Pilha : Pilha_Enc) : boolean;
23: function Tamanho (var Pilha : Pilha_Enc) : integer;
24: procedure Empilhar (Info : Tipo_do_Dado; var Pilha : Pilha_Enc);
25: procedure Desempilhar (var Info : Tipo_do_Dado; var Pilha : Pilha_Enc);
26: procedure Topo (var Info : Tipo_do_Dado; var Pilha : Pilha_Enc);
27:
28: implementation
29:
30: procedure Inicializar (var Pilha : Pilha_Enc);
31: {
32:   Objetivos: Inicializa a Pilha, atribuindo nil ao topo e zerando o
33:              tamanho da Pilha
34: }
35: begin
36:    Pilha.Topo := nil;
37:    Pilha.Tamanho := 0
38: end;
39:
40:
41: function Vazia (var Pilha : Pilha_Enc) : boolean;
42: {
43:   Objetivos: Retorna true se o numero de Nos na Pilha for igual a zero. Caso
44:              contrario, retorna false
45: }
46: begin
47:    Vazia := Pilha.Tamanho = 0;
48: end;
49:
50:
51: function Cheia (var Pilha : Pilha_Enc) : boolean;
52: {
53:   Objetivos: Retorna true se o maior bloco de memoria disponivel for menor
54:              que o tamanho de um No. Caso contrario, retorna false.
55: }
56: begin
57:    Cheia := Maxavail < SizeOf(No);
58: end;
59:
60:
61: function Tamanho (var Pilha : Pilha_Enc) : integer;
```

```
 62: {
 63:  Objetivos: Retorna o numero de Nos existentes na Pilha.
 64: }
 65: begin
 66:    Tamanho := Pilha.Tamanho;
 67: end;
 68:
 69:
 70: procedure Empilhar (Info : Tipo_do_Dado; var Pilha : Pilha_Enc);
 71: {
 72:   Objetivos: Empilha um No no topo da pilha.
 73: }
 74: var
 75:    PTemp : PNo;
 76: begin
 77:     { Testa se a Pilha esta cheia. Se estiver da uma mensagem e sai do
    programa }
 78:    if Cheia(Pilha) then
 79:       begin
 80:          writeln('ERRO: Overflow da pilha.');
 81:          halt;
 82:       end;
 83:
 84:    { Cria um novo No, coloca a informacoes nele }
 85:    New(PTemp);
 86:    PTemp^.Info := Info;
 87:
 88:    { Faz com que o novo No seja o Topo da Pilha }
 89:    PTemp^.Prox := Pilha.Topo;
 90:    Pilha.Topo := PTemp;
 91:
 92:    { Contador de Nos aumenta de um. }
 93:    inc(Pilha.Tamanho);
 94: end;
 95:
 96:
 97: procedure Desempilhar (var Info : Tipo_do_Dado; var Pilha : Pilha_Enc);
 98: {
 99:   Objetivos: Retira o No do Topo da Pilha
100: }
101: var
102:    PTemp : PNo;
103: begin
104:     { Testa se a Pilha esta vazia. Se estiver da uma mensagem e sai do
    programa }
105:    if Vazia(Pilha) then
106:       begin
107:          writeln('ERRO: Underflow da pilha.');
108:          halt;
109:       end;
110:
111:    { Coloca em Info o conteudo do No no topo da Pilha. O No que esta abaixo
112:      do Topo passa a ser o Topo da Pilha e o primeiro No, desalocado da
    memoria.
113:      Depois, o contador de Nos diminui de um. }
114:    Info := Pilha.Topo^.Info;
115:    PTemp := Pilha.Topo;
116:    Pilha.Topo := Pilha.Topo^.Prox;
117:    Dispose(PTemp);
118:    dec(Pilha.Tamanho);
119: end;
```

```
120:
121:
122: procedure Topo (var Info : Tipo_do_Dado; var Pilha : Pilha_Enc);
123: {
124:   Objetivo: Retorna o No que se encontra no Topo da Pilha sem, no entanto,
125:             retira-lo da Pilha
126: }
127: begin
128:     { Testa se a Pilha esta vazia. Se estiver da uma mensagem e sai do
     programa }
129:    if Vazia(Pilha) then
130:       begin
131:          writeln('ERRO: Topo da Pilha inexistente');
132:          halt;
133:       end;
134:
135:    { Coloca em Info o conteudo do No no topo da Pilha }
136:    Info := Pilha.Topo^.Info;
137: end;
138:
139: end.
```