```
 1: unit stack;
 2:
 3: {$mode objfpc}{$H+}
 4:
 5: interface
 6:   type
 7:     PKellerElement = ^TKellerElement; // Zeiger auf Kellerelement
 8:
 9:     TKellerElement = record           // Kellerelement mit Daten & ZEiger auf
10:       daten: integer;                 // unter ihm liegendes Element
11:       darunter: PKellerElement;       // ist dieses NIL -> letztes Element
12:     end;
13:
14:   function stack_init():PKellerElement;
15:   procedure stack_destroy(var kelleranfang:PKellerElement);
16:   procedure stack_push(var kelleranfang:PKellerElement; data: integer);
17:   function stack_pop(var kelleranfang:PKellerElement):integer;
18:
19: implementation
20:
21:   function stack_init():PKellerElement;
22:   var kelleranfang:PKellerElement;
23:   begin
24:       kelleranfang:= nil;
25:       result:= kelleranfang;
26:   end;
27:
28:   procedure stack_destroy(var kelleranfang:PKellerElement);
29:   var a: PKellerElement;
30:   begin
31:        while (kelleranfang^.darunter<>nil) do begin
32:          a:= kelleranfang;
33:          kelleranfang:= a^.darunter;
34:          dispose(a);
35:        end;
36:        dispose(kelleranfang);
37:        kelleranfang:= nil;
38:   end;
39:
40:   procedure stack_push(var kelleranfang:PKellerElement; data: integer);
41:   var neuesElement: PKellerElement;
42:   begin
43:        new(neuesElement);
44:        neuesElement^.daten:= data;
45:        neuesElement^.darunter:= kelleranfang;
46:        kelleranfang:= neuesElement;
47:   end;
48:
49:   function stack_pop(var kelleranfang:PKellerElement):integer;
50:   var a: PKellerElement;
51:     erg: integer;
52:   begin
53:        erg:= kelleranfang^.daten;
54:        a:= kelleranfang;
55:        kelleranfang:= kelleranfang^.darunter;
56:        Dispose(a);
57:        writeln(erg, ' vom STACK entfernt!');   // kann auskommentiert werden
58:        result:= erg;
59:   end;
60:
61: end.
```