```
 1: //shows the recursive solution to the 8 queens chess problem, max@kleiner.com
 2: //small output of solution in text mode or file list!, loc's =99
 3: //92 board solutions are possible! you can find the solutions in a txt file
 4:
 5: program EightQueensSolutions;
 6:
 7: type
 8:   Play = array[1..8] of boolean;
 9:   Play2 = array[1..8] of Play;
10:   RowCheck = array[1..8] of boolean;
11:   LeftDiagCheck = array[2..16] of boolean;
12:   RightDiagCheck = array[-7..7] of boolean;
13:
14: var
15:   Board : Play2;
16:   safeRow: rowCheck;
17:   safeleftDiag: LeftDiagCheck;
18:   safeRightDiag: RightDiagCheck;
19:   row, column, i, tester: integer;
20:   reslist: TStringList;
21:
22:
23: function Safe(row,col: byte; saferow: rowCheck; safeleftDiag: leftDiagCheck;
24:                       saferightDiag: rightDiagCheck): boolean;
25: //true if a queen can be safely placed in the current position
26: begin
27:  result:= safeRow[row] and safeLeftDiag[row+col] and safeRightDiag[row-col]
28: end;
29:
30: procedure chessPrint(board: Play2);
31: var i,j: integer;
32:     tempstr, codestr: string;
33: //shows the current board layout
34: begin
35:   tempstr:= '';
36:   codestr:= '';
37:   reslist.add(inttostr(tester+1))
38:   //i is column , j is row !
39:   for j:= 1 to 8 do begin
40:     for i:= 1 to 8 do
41:       if (board[i][j]) then begin
42:         tempstr:= tempstr + 'Q'
43:         codestr:= codestr + inttostr(i)
44:       end else tempstr:= tempstr + 'X';
45:         reslist.add(tempstr)
46:         tempstr:= '';
47:         //write('Q') else write('*');
48:         //writeln('');
49:   end; //for
50:   reslist.add(codestr);
51:   reslist.add('-----------------');
52:   inc(tester)
53:   writeln(inttostr(tester) + ': '+codestr);
54: end;
55:
56: procedure TryColumn(column: integer; var board: Play2);
57: //recursive procedure for attempting queen placement
58: var row: integer;
59: begin
60:   row:= 1
61:   repeat
62:     if safe(row, column, saferow, safeleftdiag, saferightdiag) then begin
63:       //set the queen
64:       saferow[row]:= false;
65:       safeleftdiag[row+column]:= false;
66:       saferightdiag[row-column]:= false;
67:       board[row][column]:= true;
68:       if column < 8 then
69:         Trycolumn(column + 1, board) else
70:            chessPrint(board);
71:       //removes queens
72:       safeRow[row]:= true;
73:       safeleftDiag[row+column]:= true;
74:       saferightDiag[row-column]:= true;
75:       board[row][column]:= false;
```

```
76:     end;
77:     //row was safe
78:     inc(row);
79:   until row > 8
80: end;
81:
82: begin
83:   //init board & main
84:   tester:= 0;
85:   reslist:= TStringlist.create;
86:   for row:= 1 to 8 do safeRow[row]:= true;
87:   for i:= 2 to 16 do safeleftDiag[i]:= true;
88:   for i:= -7 to 7 do saferightDiag[i]:= true;
89:   for row:= 1 to 8 do
90:     for column:= 1 to 8 do
91:       board[row][column]:= false;
92:       //make the first recursive call
93:     tryColumn(1, board);
94:     reslist.savetoFile('chesssolution_result.txt')
95:     memo2.lines.add('all codestrings of solutions');
96:     reslist.free;
97: end.
```