Das Springerproblem

1 Einführung

Das Springerproblem ist ein bekanntes Problem zur Demonstration des sogenannten Backtracking-Verfahrens, einem rekursiven Lösungsverfahren für Suchprobleme.

Problemstellung

Die Figur des Springers wird auf ein beliebiges Feld eines Schachbretts gestellt. Gesucht ist eine Zugfolge des Springers, bei der der Springer nach den Schachregeln über das Brett springt und dabei jedes Feld einmal betritt.

Lösungsalgorithmus

Man bewegt den Springer probeweise über das Schachbrett und notiert sich dabei, welche Felder schon besucht wurden. Sollte der Springer auf ein Feld gelangen, von dem aus es nicht möglich ist, ein unbesuchtes Feld zu erreichen, dann werden der oder die letzten Züge zurückgenommen, bis eine Position erreicht ist, von der aus der Springer wieder mindestens ein unbesuchtes Feld erreichen kann.
So wird immer weiter verfahren, bis eine Zugfolge ermittelt wurde, mit der alle Felder besucht wurden.

```
procedure "versuche den nächsten Zug"
  begin
    repeat
     "wähle nächsten Zug aus der Liste möglicher Züge aus"
     if "korrekter Zug"
       then begin
            "zeichne Zug auf"
            if "Brett nicht voll" then
                                  begin
                                   "versuche den nächsten Zug"
                                   if not "erfolgreich"
                                   then "lösche vorangegangene Aufzeichnung"
                                  end;
            end;
   until "erfolgreich" or "keine Züge mehr möglich"
  end;
```

Ein Beispiel

Die folgende Abbildung zeigt eine Zugfolge des Springers auf einem 5*5-Feld ausgehend von dem Feld in der linken oberen Ecke. Die Züge sind durchnummeriert.

1	6	15	10	21
14	9	20	5	16
19	2	7	22	11
8	13	24	17	4
25	18	3	12	23

2 Implementierung

Das Folgende Programm ist eine Implemetierung in PASCAL, bei der die Größe des Schachbretts über die Konstante *n* bestimmt werden kann. Das Programm arbeitet mit einer Sprungtabelle, die die gültigen Züges des Springers nach den Schachregeln gewährleistet.

```
program springerproblem;   {Backtracking - ein rekursives Verfahren}
uses wincrt;
{ Das folgende Programm implementiert das Springerproblem aufgrund
  der folgenden Prozedur:

 procedure "versuche den nächsten Zug"
 begin
   repeat
    "wähle nächsten Zug aus der Liste möglicher Züge aus"
    if "korrekter Zug"
      then begin
           "zeichne Zug auf"
           if "Brett nicht voll" then
                                 begin
                                  "versuche den nächsten Zug"
                                  if not "erfolgreich"
                                  then "lösche vorangegangene Aufzeichnung"
                                 end;
           end;
  until "erfolgreich" or "keine Züge mehr möglich"
 end;
}

{Zur Implementation werden folgende Datentypen benötigt}
const n=5;  {Für ein Brett mit 8x8 Feldern}

type
   brett=Array[1..n,1..n] of integer;  { Das Schachbrett: }
                             { Eintrag 0 => Feld  wurde noch nicht besucht,}
                             { Eintrag i => Feld wurde im i-ten-Zug besucht}

var Schachbrett:brett;
    s:set of 1..n;
    a,b:Array [1..8] of integer; {Tabelle für Sprungweiten}
    wiewars:boolean;

procedure DruckeSchachbrett;
var i,j:integer;
begin
for i:=1 to n do begin
  for j:=1 to n do begin
    write(' | ',Schachbrett[i,j]);
    end;
 writeln;
 end;
end;
```

```
procedure initialisiereSprungtabelle;
begin
s:=[1,2,3,4,5];
a[1] := 2;     b[1]:= 1;
a[2] := 1;     b[2]:= 2;
a[3] := -1;    b[3]:= 2;
a[4] := -2;    b[4]:= 1;
a[5] := -2;    b[5]:= -1;
a[6] := -1;    b[6]:= -2;
a[7] := 1;     b[7]:= -2;
a[8] := 2;     b[8]:= -1;
end;

procedure initialisiereSchachbrett;
var i,j:integer;
begin
for i:=1 to n do
 for j:=1 to n do
  Schachbrett[i,j]:=0;  {Feld noch nicht besucht}
end;

procedure try(ZugNr:integer; x,y:integer ; VAR q:boolean);
var k:integer; {Kandidatennummer}
    u,v:integer; {Indizes von Feld [u,v]}
    erfolgreich:boolean;
begin
 k:=0;
 repeat
 k:=k+1; {nächste Kandidatennummer}
 erfolgreich:=false; {bisher noch nicht erfolgreich}
 u:=x+a[k];
 v:=y+b[k]; {bestimmt den nächsten Feld-Kandidaten [u,v] anhand der
Sprungtabelle}
 if (u in s) and (v in s) {korrekter Zug}
   then if Schachbrett[u,v] = 0
           then
           begin
           Schachbrett[u,v]:=ZugNr;
           if (ZugNr < n*n)  {noch nicht alle Züge gemacht}
              then
              begin
              try(ZugNr+1,u,v,erfolgreich);
              if not erfolgreich then  Schachbrett[u,v]:=0; {nimm Zug
zurück}
              end
              else erfolgreich := true;
           end;
 until erfolgreich or (k=8);
 q:=erfolgreich;
 end; {of try}


begin{Hauptprogramm}
      initialisiereSprungtabelle;
      initialisiereSchachbrett;
      DruckeSchachbrett;
      Schachbrett[1,1]:=1; {Startfeld schon mal auf 1 setzen}
      try(2,1,1,wiewars);
      if wiewars=true then writeln('erfolgreich!')
                      else writeln('nicht erfolgreich!');
      DruckeSchachbrett;
end.{Hauptprogramm}
```