probiert wurden, :gelegt die angehende Lösung und :vorbei die Karten im Papierkorb. Zuerst wird geprüft, ob alle neun Karten gelegt wurden [6], dann wäre eine Lösung gefunden und könnte ausgegeben werden [7]. Dann wird geprüft, ob noch Karten im Vorrat sind [8]. Wenn nicht, geht die Programmausführung um eine Rekursionsstufe zurück [12/14]. Ein derartiges Abbruchkriterium [8] muss in jeder rekursiven Prozedur enthalten sein.

In der aktuellen Ausgabe von suchen wird versucht, die erste Karte aus :vorrat zu legen: Deren Beschreibung wird der Variablen karte zugewiesen [9]. Die LOCAL-Anweisung [5] sorgt dafür, dass :karte auf jeder Stufe der Rekursion separat erhalten bleibt. Kann :karte gelegt werden [11], so beginnt suchen wieder von vorne, mit dem neuen :vorrat bestehend aus den Karten in Papierkorb (:vorbei) und dem alten :vorrat. Zu der Liste :gelegt mit den erfolgreich gelegten Karten wird die :karte hinzugefügt; probiert wurde noch keine Karte, also ist :vorbei leer. - Nach der Rückkehr aus der Rekursion wird die :karte um 90° gedreht [13,26].

Die Anweisungsliste der REPEAT-Anweisung wird viermal ausgeführt, um alle möglichen Drehrichtungen zu erhalten, ausser wenn die Karte in Zentrum gelegt wird [10]. Dank diesem Trick findet das Programm nur die echt verschiedenen Lösungen, und alle, die daraus durch Drehung entstehen, werden unterdrückt.

Nachdem mit dieser Karte alle Möglichkeiten durchgespielt wurden, ruft suchen sich selbst wieder auf, wobei :karte aus :vorrat entfernt und in :vorbei abgelegt wird. Hier handelt es sich nur der Form nach um eine Rekursion, denn die aktuelle Situation interessiert ja nach der Rückkehr nicht mehr: es bleibt nur noch die END-Anweisung. Obwohl eine (Endlos-)Schleife den gleichen Effekt hätte, ist es in LOGO üblich, die elegantere, rekursive Formulierung zu verwenden. Gute LOGO-Implementationen sind ohnehin in der Lage, diesen Spezialfall zu erkennen und durch eine Iteration zu ersetzen.

Die Funktion testkarte [15] prüft, ob :karte zu :gelegt passt. Wenn noch gar keine Karte gelegt ist, passt sie sowieso [16]. Nur wenn nicht genau drei oder genau sechs Karten gelegt sind [17], muss sie gegen links passen [18]. Wenn mindestens drei Karten schon gelegt sind [19], muss sie auch gegen oben passen [20]. Ist sie bei keinem der Tests durchgefallen, so passt sie [21].

Die Funktion test2 [22] erhält als Argumente die Beschreibungen zweier Schildkrötenhälften, z.B. "f2 und "f1, und prüft, ob diese zusammenpassen. (Hier wäre das der Fall.)

Die mod-Funktion [23], die den Divisionsrest liefert, wird natürlich rekursiv programmiert.

Die Funktion drehe [24] führt in einem recht komplizierten Ausdruck die Operation einer 90°-Drehung einer Karte durch.