Jugendwettbewerb Informatik 2019

Runde 3

Junioraufgabe 2: Kacheln

25. November 2019

Inhaltsverzeichnis

Lösungsidee

Die Kacheln sind aus 4 Quadraten zusammengesetzt (4-Teilig). Beim Zusammensetzen von Landschaften soll eine Regel beachtet werden. Die Außenseiten der Quadrate dürfen nur so aneinander gelegt werden, dass Land auf Land und Wasser auf Wasser trifft. Land wird durch 1 und Wasser durch 0 dargestellt.

Das Programm muss also die Außenseiten der Quadrate einer Kachel mit den Außenseiten der Quadrate von Nachbarkacheln vergleichen, um festzustellen, ob ein Anlegen möglich ist.

Es darf nur angelegt werden, wenn die Beschaffenheit (1 oder 0) vom Quadrat der Kachel mit den Vergleichsquadraten auf den anderen Kacheln übereinstimmt.

Es sollen unvollständige Kachel-Landschaften vervollständigt werden. Die zu bestimmenden Quadrate sind mit * gekennzeichnet.

Mit welchen Elementen verglichen wird, hängt ab von wo das Quadrat auf der Kachel liegt. Es gibt 4 Typen.

- oben links → Vergleichselemente: links, oben, links-oben

- oben rechts → Vergleichs-Elemente: rechts, oben, rechts-oben

- unten links → Vergleichs-Elemente: links, unten, links-unten

- unten rechts → Vergleichs-Elemente: rechts, unten, rechts-unten

Nachdem die jeweiligen Vergleichselemente in der Kachel-Landschaft gefunden wurden, muss geschaut werden, ob ein vervollständigen ohne Regelverletzung möglich ist.

Angrenzende Vergleichselemente dürfen entweder unbekannt (*) sein oder wenn bekannt, müssen die bekannten alle von der selben Art sein (alle bekannten Vergleichselemente sind 1, oder alle bekannten sind 0).

Gibt es sowohl Vergleichselemente die 0 als auch welche, die 1 sind, dann kann nicht vervollständigt werden (Regelverletzung) und es soll eine entsprechende Meldung ausgegeben und das Programm beendet werden.

Sonst soll das Quadrat bestimmt werden als:

- 0 wenn mindestens ein Vergleichselement 0 oder alle *

- 1 wenn mindestens ein Vergleichselement 1

Zum Schluss soll die vervollständigte Landschaft ausgegeben werden.

Umsetzung

Das Programm habe ich mit Blocky geschrieben, da ich leider noch keine andere Programmiersprache kann.

Ich habe die zur Verfügung gestellte Eingabe-/Ausgabe Umgebung verwendet. Das Programm sind also die gespeicherten Blöcke als Datei (Kacheln), die in die Umgebung geladen werden müssen. Die Eingaben müssen aus den vorgegebenen Textfiles in der kompakten Version (z.b. map compact.txt) in das Eingabefeld kopiert werden.

Bitte nicht vergessen, die Eingabe mit der „enter“ Taste abzuschließen, da sonst die letzte Zeile nicht eingelesen wird.

Das Programm liest mit der Funktion einlesen zuerst die Anzahl der Kachel-Zeilen und Kachel-Spalten ein und speichert sie in den Variablen (zeilen, spalten). Dann wird die unvollständige Kachel-Landschaft zeilenweise eingelesen und in einer Liste aus Listen (kachel_liste) abgelegt.

Die Funktion rand fügt so zusätzliche Elemente in die kachel_liste ein, dass ein Rand um die Landschaft entsteht. Der Rand macht Elemente-Vergleiche bei Randkacheln einfacher.

Im Hauptteil des Programms wird die zu vervollständigende Kachellandschaft Quadrat für Quadrat nach unbekannten Quadraten (*) durchsucht. Dies erledigt eine geschachtelte Schleifenstuktur.

Wird ein * gefunden, wird, mit der Funktion typ_bestimmen, ermittelt, wo sich das Quadrat auf der Kachel befindet (links oben=Typ1, rechts oben = Typ2, links unten = Typ3, rechts unten = Typ4).

Die Funktionen typ1, typ2, typ3 und typ4 finden dann die für den Typ relevanten Vergleichselemente und die Funktion vergleichen, wertet die Vergleichselemente aus. Gibt es unter den Vergleichselementen sowohl 1 als auch 0 wird ein Fehlercode („X“) ausgegeben. Sonst, gibt es bei den bekannten Vergleichselementen eine 1, dann wird 1 zurückgegeben, sonst 0.

Wurde beim Vergleichen ein Fehlercode generiert, dann bricht das Programm ab und gibt aus: „Es ist nicht möglich die Kachellandschaft zu vervollständigen!!!“

Sonst wird das neu bestimmte Element in die Kachellandschaft (kachel_liste) eingefügt.

Konnte die Landschaft vervollständigt werden, wird sie zum Schluss ausgegeben (ausgeben).

Zuvor entfernt die Funktion randweg den Hilfsrand wieder.

Beispiele

Vorgegebene Beispiele

Das hat mein Programm für die vorgegebenen Beispiele ausgegeben.

Beispiel 1

Eingangsdatei: map compact.txt

Ausgangslandschaft:

```
10**
01**
**10
**11
```

Vervollständigte Landschaft

```
1000
0110
0110
0111
```

Beispiel 2

Eingangsdatei: map1 compact.txt

Ausgangslandschaft:

```
1001****111001
0110****011001
0110******1001
1110******1111
****000001****
****100001****
000110**0111**
011111**1110**
01111111111000
11100000011001
1110******1001
0111******0001
```

Vervollständigte Landschaft

```
10011001111001
01100000011001
01100000011001
11100000011111
11100000011111
00011000011110
00011000011110
01111111111000
01111111111000
11100000011001
11100000011001
01111000000001
```

Junioraufgabe 2: Parallelen

Beispiel 3

Eingangsdatei: map2 compact.txt

Ausgangslandschaft:

```
******
******
**00**
**10**
**10**
**00**
******
******
```

Vervollständigte Landschaft

```
000000
000000
000000
011000
011000
000000
000000
000000
```

Beispiel 4

Eingangsdatei: map3 compact.txt

Ausgangslandschaft:

```
10000111100001
11111001111000
1111********00
0001********11
0001********11
1110********01
11100110000001
11111110000111
```

Vervollständigte Landschaft

```
10000111100001
11111001111000
11111001111000
00011000000111
00011000000111
11100110000001
11100110000001
11111110000111
```

Beispiel 5

Eingangsdatei: map4 compact.txt

Ausgangslandschaft:

```
11100000011001111001111000011111
00011001111000011111100000011001
00011001111000011111100000011001
01111111111110000000011001111000
01111111******000000011001111000
11100000******011001111001111000
11100000******011001111001111000
00000000******111111111110000000
00000000000111111111111110000000
00000000011111111110000001111001
00000000011111111110****01111001
00011001111110000110****11100000
00011001**1110000110****11100000
00011001**0000000000****11111111
00011001****00000000011111111111
01111000****00000001111110000000
01111000******000001111110000000
00000000******000000000000000000
```

Vervollständigte Landschaft

```
11100000011001111001111000011111
00011001111000011111100000011001
00011001111000011111100000011001
01111111111110000000011001111000
01111111111110000000011001111000
11100000000000011001111001111000
11100000000000011001111001111000
00000000000111111111111110000000
00000000000111111111111110000000
00000000011111111110000001111001
00000000011111111110000001111001
00011001111110000110000111100000
00011001111110000110000111100000
00011001100000000000011111111111
00011001100000000000011111111111
01111000000000000001111110000000
01111000000000000001111110000000
00000000000000000000000000000000
```

Beispiel 6

Eingangsdatei: map5 compact.txt

Ausgangslandschaft:

```
1111111111
1000000111
100000**11
000110**00
**01**1000
**00**0000
**0001**00
**1111**11
0111111111
1110000111
```

Programmausgabe:

```
Es ist nicht moglich
die Kachellandschaft
zu vervollstandigen!!!
```

Eigene Beispiele

Beispiel 7

Eingangsdatei: eigenes_beispiel1.txt

Ausgangslandschaft:

```
**
**
```

Programmausgabe:

```
00
00
```

Wenn für ein Element sowohl Land als auch Wasser möglich wäre, wählt mein Programm immer Wasser (0). Das führt hier zu einer reinen Wasser-Landschaft.

Beispiel 8

Eingangsdatei: eigenes_beispiel2.txt

Ausgangslandschaft:

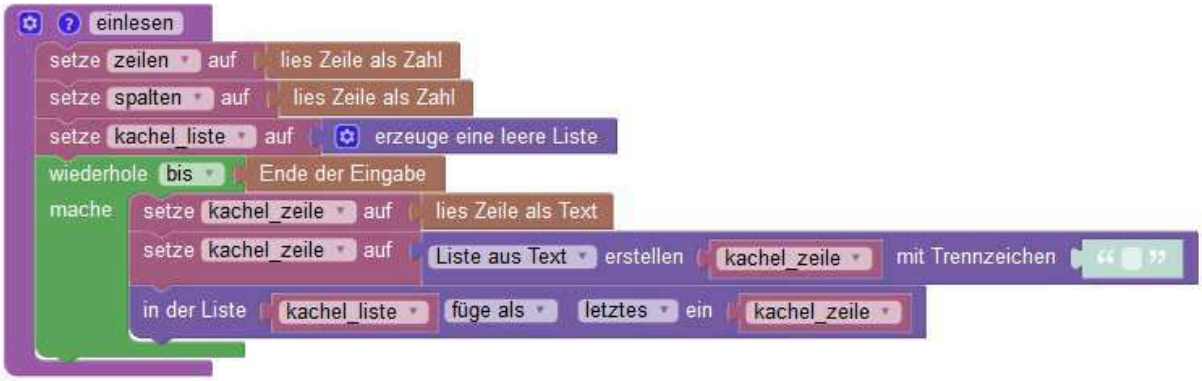
```
**11
11**
```

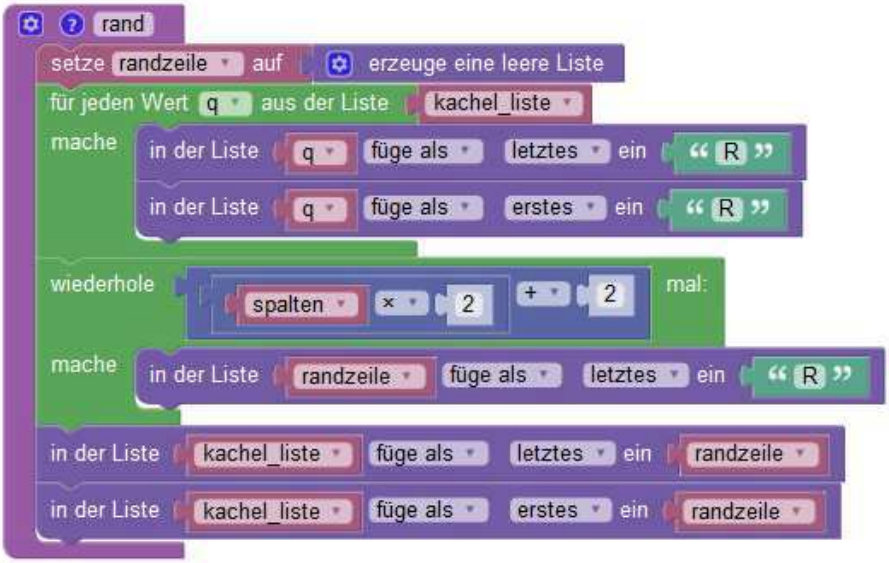
Programmausgabe:

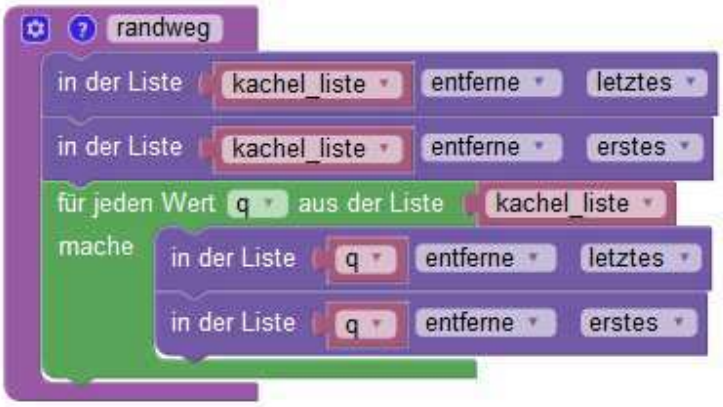
```
0111
1110
```

Quellcode

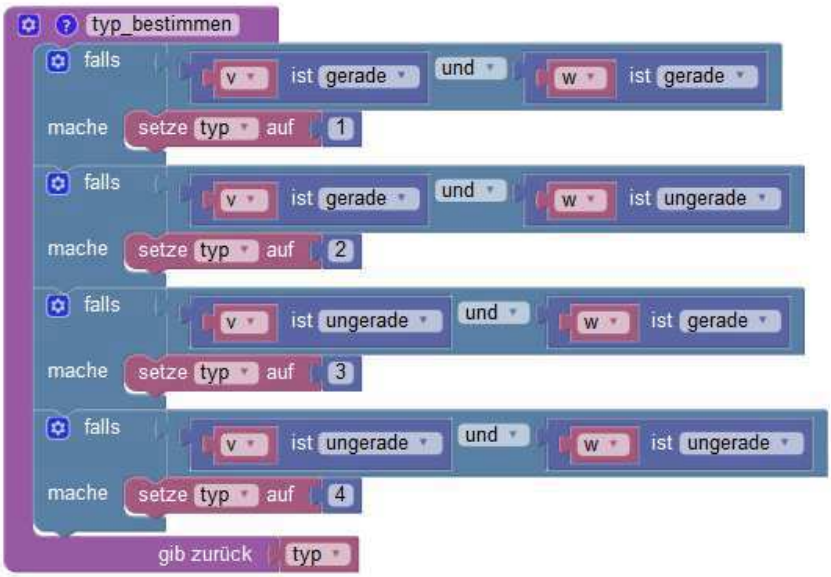
Hier die Blöcke von meinem Blockly – Programm. Sie sind in der Datei Kacheln gespeichert.

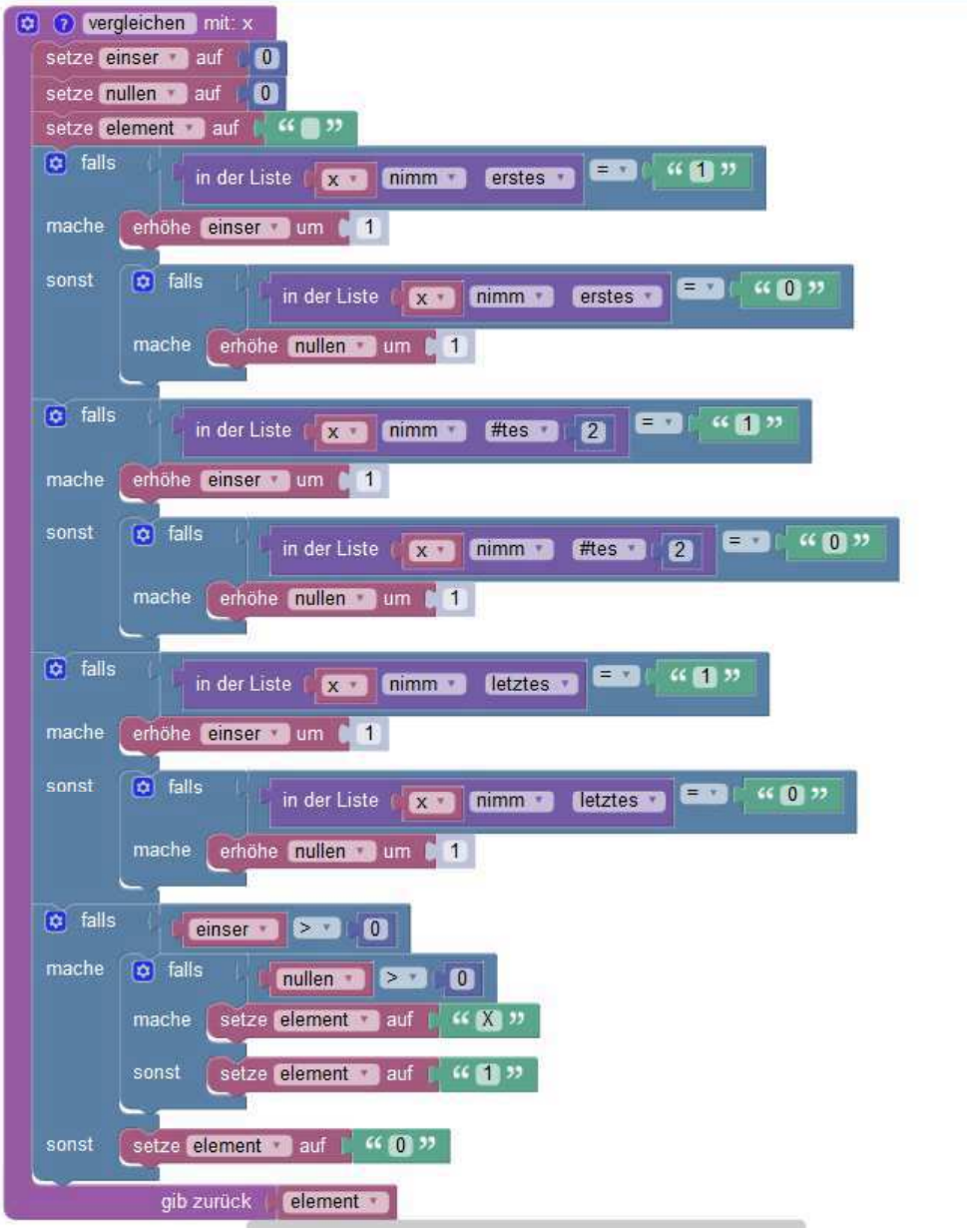




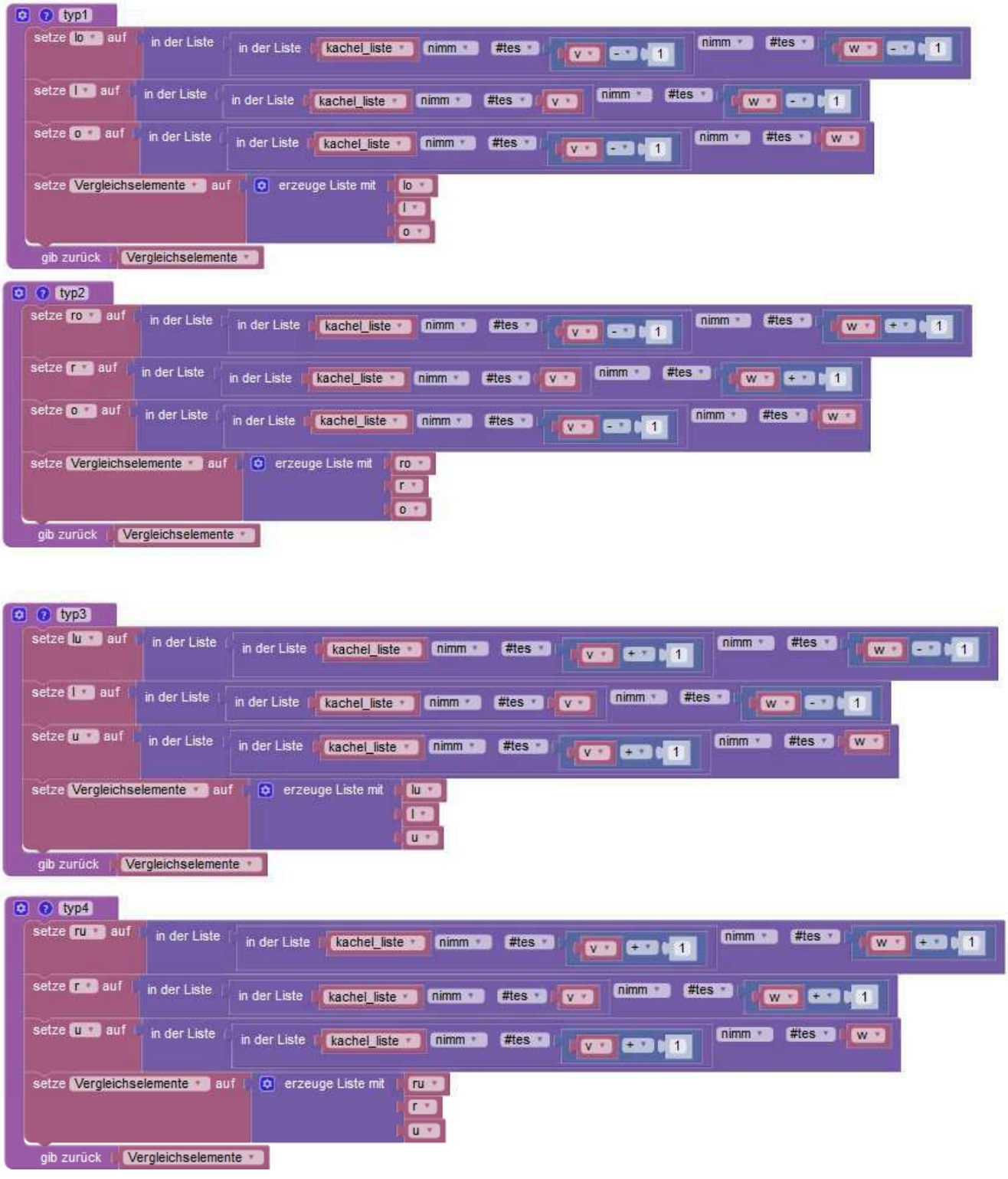


Junioraufgabe 2: Parallelen





Junioraufgabe 2: Parallelen



Junioraufgabe 2: Parallelen