```go
// Definieren einer 2D-Slice
var spielfeld [][]string
```

Zu Anfang leere Liste von Zeilen

Zugrundeliegendes Array

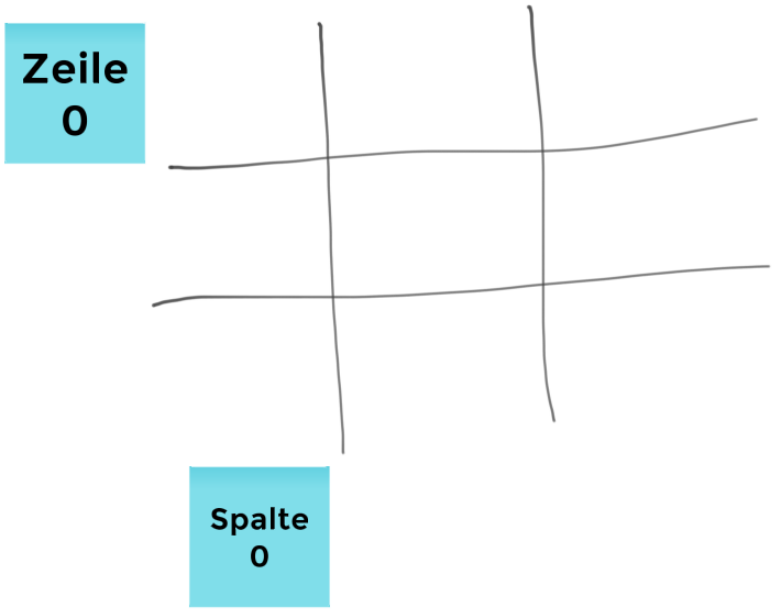
Slice "spielfeld"

```go
func makeBoard(size int, initChar string) [][]string {
  // Definieren einer 2D-Slice
  var board [][]string

  for i := 0; i<size; i++ {
    var row []string

    board = append(board, row)
  }

  return board
}
```

Liste von size leeren Zeilen

Zugrundeliegendes Array

Slice "spielfeld"



Ein Spieler hat gewonnen, wenn

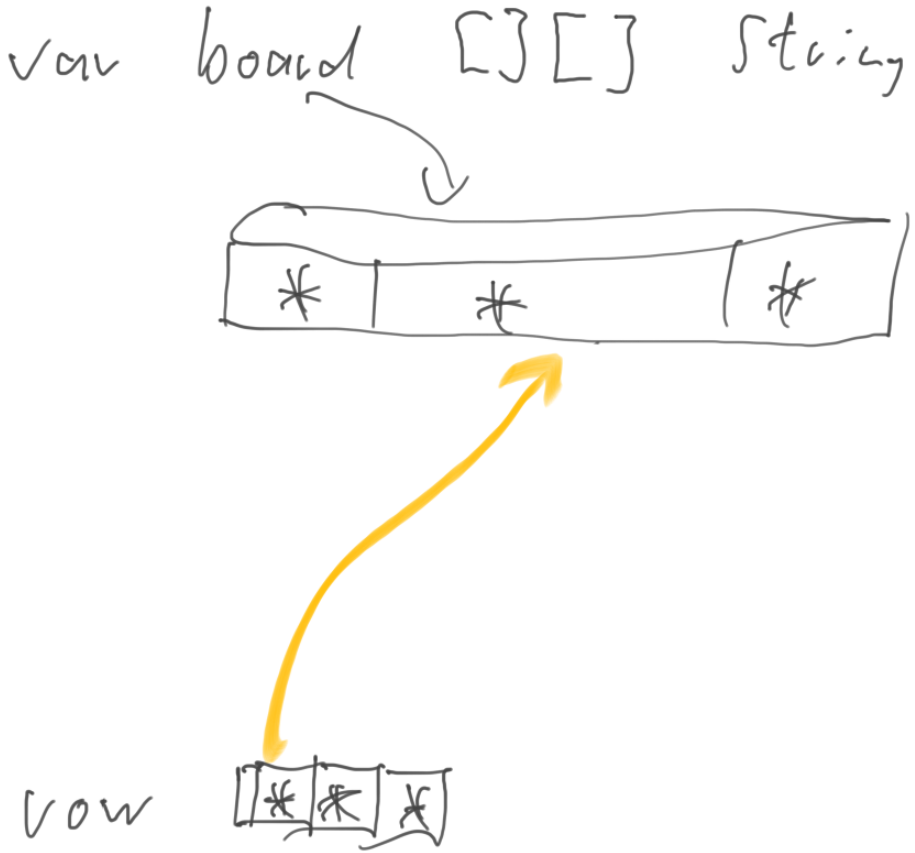
er eine Zeile, Spalte oder Diagonale voll hat.

- eine Zeile ist voll
- eine Spalte ist voll
- eine Diagonale ist voll

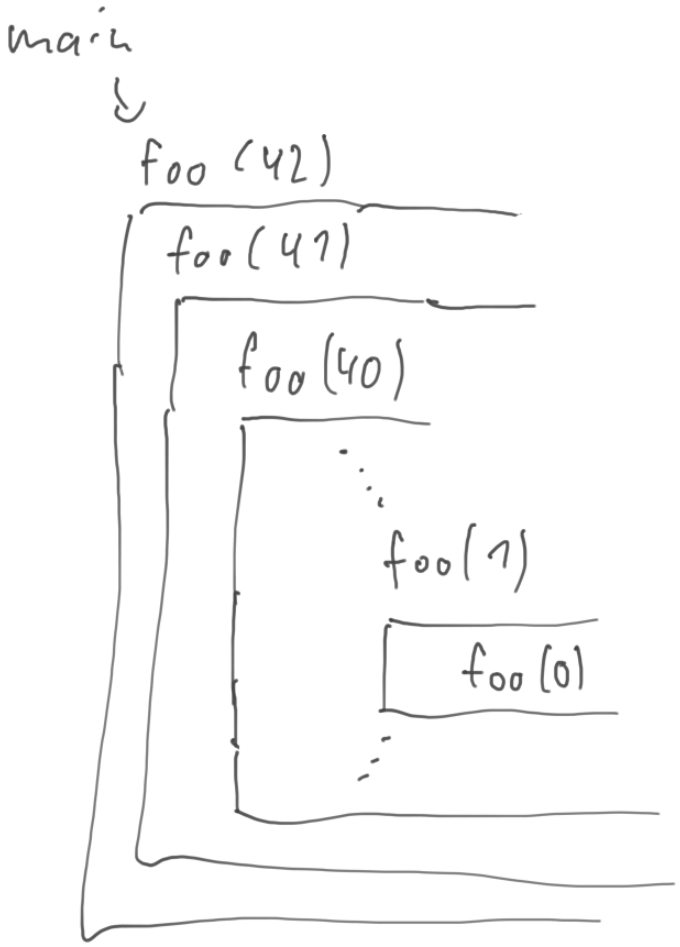
- Zeile 0 ist voll oder Zeile 1 ist voll oder Zeile 2 ist voll
- Spalte 0 ist voll oder Spalte 1 ist voll oder Spalte 2 ist voll
- ...

var board [][] string

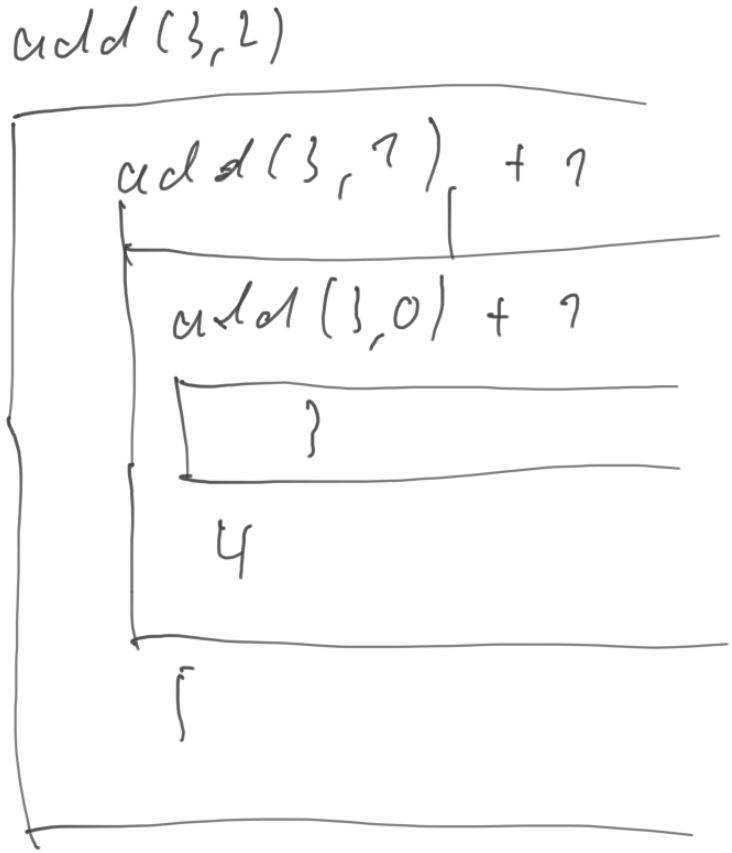
```go
func makeBoard(size int, initChar string) [][]string {
  // Definieren einer 2D-Slice
  var board [][]string

  for i := 0; i<size; i++ {
    var row []string
    for j := 0; j < size; j++ {
      row = append(row, initChar)
    }
    board = append(board, row)
  }
  return board
}
```



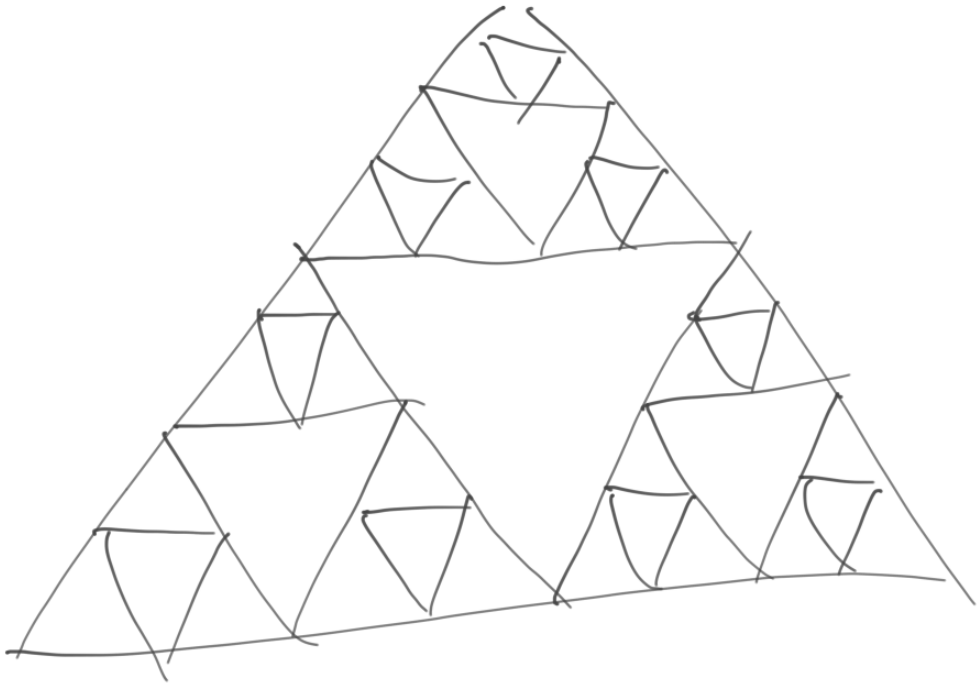

```go
func makeBoard(size int, initChar string) [][]string {
  // Definieren einer 2D-Slice
  var board [][]string

  for i := 0; i<size; i++ {
    var row []string
    for j := 0; j < size; j++ {
      row = append(row, initChar)
    }
    board = append(board, row)
  }
  return board
}
```




```
func foo(k int) {
  if k == 0 {
    return
  }
  fmt.Println(k)
  foo(k - 1)
}
```




```
func add(x, y int) int {
  if y == 0 {
    return x
  } else {
    return add(x, y-1) + 1
  }
}
```



$$A(m,n) = \begin{cases} n+1 & \text{falls } m = 0 \\ A(m-1,1) & \text{falls } m > 0 \text{ und } n = 0 \\ A(m-1, A(m,n-1)) & \text{falls } m > 0 \text{ und } n > 0 \end{cases}$$

$A(1,1) = A(0, A(1,0))$

$= A(0, 1)$

$m \backslash n$	0	1	2	3	...
0	1	2	3	4	...
1	2	3			
2	3				
3	.				
⋮	⋮