Opdracht 'doolhof'

Maak een application die in staat is van een gegeven doolhof de uitgang te vinden. Je krijgt de beschikking over een aantal methoden en classes die je in staat stellen een afbeelding van een doolhof in te lezen. Wat je zelf nog te doen hebt is de methode Solve() in te vullen in de class Maze (bestand 'Maze.cs').

Ga uit van de opdracht:
"Vindt een weg naar de uitgang door vanuit een (begin) cel alle mogelijke wegen te bewandelen". Uiteraard mag de begin-cel niet als uitgang worden gezien. Zou je al zoekend weer in de begin-cel terechtkomen, dan dient die mogelijkheid te worden overgeslagen.

"Ga vanuit de 'huidige cel' alle mogelijkheden na om een nieuwe cel te betreden welke niet voorkomt in een lijst van reeds betreden cellen".
Elke keer dat je dat doet dient die cel als 'huidige cel' te worden gezien en heb je eigenlijk dezelfde opdracht, maar dan ben je wel één cel verder opgeschoten.

Tijd dus voor een recursieve methode!

Wat is hier het "stop-criterium"?.

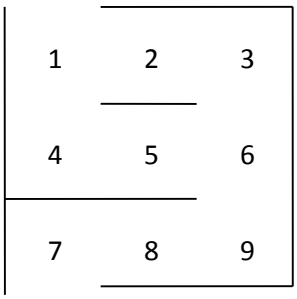
Simpel: de nieuw te onderzoeken cel ligt buiten het doolhof.

Waar moet je rekening mee houden?

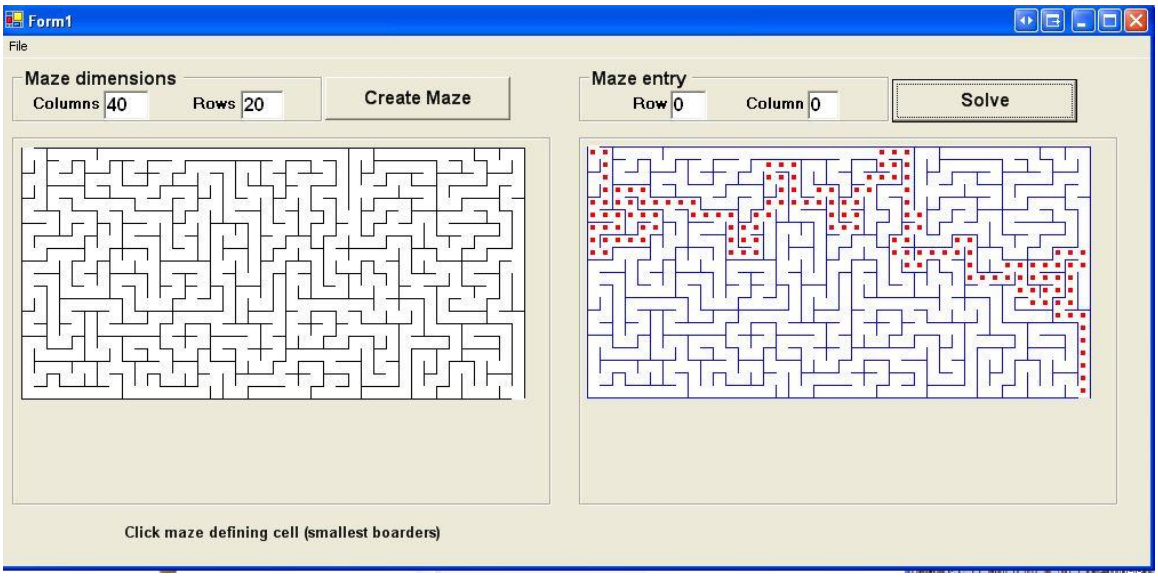
Dat je niet een cel opnieuw onderzoekt waar je al zoekende reeds geweest bent.

Dit betekent dat je elke keer als je een nieuwe cel als huidige cel neemt (en dus een recursieve aanroep naar jezelf doet), je deze cel toevoegt aan een array. Het level van de aanroep is de positie van deze cel in het array, de waarde is het nummer van de cel.

de lijst met bezochte vakjes:
reedsbezocht[1] = 1
reedsbezocht[2] = 4
reedsbezocht[3] = 5
reedsbezocht[4] = 6
reedsbezocht[5] = 3
reedsbezocht[6] = 2
reedsbezocht[7] = 9
etc …



Je maakt gebruik van een omhullend programma wat al is gegeven. Hierbinnen ben je in staat een .jpg van een doolhof in te laden, en de gevonden oplossing te laten tekenen.



Binnen het programma wordt de definitie van een doolhof opgeslagen in een matrix
private int[,] cellMatrix; dus van int[<rijen>,<kolommen>]
Per cel van het doolhof wordt hierin m.b.v. een getal aangegeven welke kanten de cel "open" heeft:
//cell filled with:

```
//0 : all sides free
//1 : left closed
//2 : top closed
//4 : right closed
//8 : bottom closed
//else: combination of sides closed (max=15 : all sides closed)
```

Aanwijzingen:

- Maak gebruik van het array private int[,] cellMatrix
- Maak gebruik van een array int[] pad (zie attributes in de class Maze) waarin de cellen van het doolhof liggen opgeslagen in de vorm van de positie van de cel:

r * squareSizeW + c. Hierin is squareSizeW dan het aantal cellen per rij (aantal kolommen dus).

- Maak gebruik van de bool solutionFound om als stop-criterium te gebruiken
- Bedenk dat het vinden van de oplossing inhoudt dat je je aan de rand bevindt van het doolhof. Maar bij aanvang van het zoekproces ook.

Om te zien welke zijden open zijn moet je de waarde in een cel (in 2-dim array 'cellMatrix') gebruiken.