# Sudoku

Generated by Doxygen 1.9.7

# Chapter 1

# Class Index

## 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 2

# File Index

## 2.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 3

# Class Documentation

## 3.1 Sudoku Class Reference

`#include <Sudoku.hpp>`

### Public Member Functions

- Sudoku (string)

  *Liest ein Sudoku-Feld aus einer CSV-Datei ein.*
- Sudoku (int f[9][9])

  *Spielfeld wird ueber ein zweidimensionales Array eingelesen.*
- void solve ()

  *Loesst das Sudoku mit einem Backtracking Algorithmus.*
- void **printField** ()

  *Gibt das Feld auf der Konsole aus.*
- bool checkSudoku ()

  *Prueft das komplette Sudoku auf Korrektheit in allen Zeilen, Spalten und Bloecken.*

### Private Member Functions

- bool checkForZeroes ()

  *Ueberprueft das Feld auf Nullen (Spiel nicht geloesst)*
- bool checkForMistakes ()

  *Ueberprueft ob die Zahlen im Feld ausserhalb {0,...,9} liegen.*
- bool checkBlock (int, int, int, int)

  *Ueberprueft ob eine Zeile, Spalte oder ein Block korrekt ist.*

### Private Attributes

- int field [9][9]
- bool solved
- bool zeroes
- bool mistake

### 3.1.1 Detailed Description

**Author**

benjamind

**Date**

7/17/2023

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 Sudoku() [1/2]

```
Sudoku::Sudoku (
            string csvPath )
```

Liest ein Sudoku-Feld aus einer CSV-Datei ein.

**Parameters**

| | |
|---|---|
| *csvPath* | Pfad an dem die CSV-Datei gespeichert ist. |

#### 3.1.2.2 Sudoku() [2/2]

```
Sudoku::Sudoku (
            int f[9][9] )
```

Spielfeld wird ueber ein zweidimensionales Array eingelesen.

**Parameters**

| | |
|---|---|
| *f* | Array (9x9) welches das Spielfeld enthält. |

### 3.1.3 Member Function Documentation

#### 3.1.3.1 checkBlock()

```
bool Sudoku::checkBlock (
            int rowStart = 0,
            int colStart = 0,
            int rowEnd = 9,
            int colEnd = 9 )  [private]
```

Ueberprueft ob eine Zeile, Spalte oder ein Block korrekt ist.

Es muessen alle Ziffern von 1 bis 9 in den (3x3, 1x9, 9x1) Matritzen enthalten sein. Zum pruefen wird die Summe der quadrate (mit 285) verglichen.

**Parameters**

| | |
|---|---|
| *rowStart* | Reihe in der gestartet wird |
| *colStart* | Spalte in der gestartet wird |
| *rowEnd* | Reihe bis zu der Ueberprueft wird |
| *colEnd* | Spalte bis zu der Ueberprueft wird |

**Returns**

Liefert ¨true¨ falls alle Ziffern enthalten sind.

#### 3.1.3.2 checkForMistakes()

```
bool Sudoku::checkForMistakes ( )  [private]
```

Ueberprueft ob die Zahlen im Feld ausserhalb {0,...,9} liegen.

ist dies der Fall, wird das Attribut ¨mistake¨ auf ¨true¨ gesetzt

**Returns**

Liefert ¨True¨ falls ein solcher Fehler gefunden wird.

#### 3.1.3.3 checkForZeroes()

```
bool Sudoku::checkForZeroes ( )  [private]
```

Ueberprueft das Feld auf Nullen (Spiel nicht geloesst)

Sind Nullen im Feld enthalten, werden die Attribute ¨zeroes¨ auf ¨true¨ und ¨solved¨ auf ¨false¨ gesetzt.

**Returns**

Liefert ¨True¨ zurueck wenn Nullen im Feld sind.

#### 3.1.3.4 checkSudoku()

```
bool Sudoku::checkSudoku ( )
```

Prueft das komplette Sudoku auf Korrektheit in allen Zeilen, Spalten und Bloecken.

Die Ueberpruefung erfolgt mithilfe der ¨checkBlock¨ Methode. Setzt das Attribut ¨solved¨ auf true wenn das Sudoku geloest ist

**Returns**

Liefert ¨true¨ falls das Sudoku korrekt geloest ist.

#### 3.1.3.5 solve()

```
void Sudoku::solve ( )
```

Loesst das Sudoku mit einem Backtracking Algorithmus.

Der Algorithmus stammt aus der SudokuBacktrack Klasse.

### 3.1.4 Member Data Documentation

#### 3.1.4.1 field

```
Sudoku::field  [private]
```

Hier ist das Spielfeld in einer 9x9 Matrix gespeichert. Es darf alle Ziffern von 0-9 enthalten. Die 0 Stellt ein Leeres (zu fuellendes) Feld dar.

#### 3.1.4.2 mistake

```
Sudoku::mistake  [private]
```

Wird durch die Methode ¨checkSudoku¨ gesetzt und gibt an ob fehlerhafte Zahlen im Feld stehen.

#### 3.1.4.3 solved

```
Sudoku::solved  [private]
```

Wird durch die Methode ¨checkSudoku¨ gesetzt und gibt an ob das Spiel geloest ist.

#### 3.1.4.4 zeroes

```
Sudoku::zeroes  [private]
```

Wird durch die Methode ¨checkSudoku¨ gesetzt und gibt an ob das Spiel nullen im Feld hat.

The documentation for this class was generated from the following files:

- C:/Users/benja/Desktop/CSE/PROG/Projekt_Sudoku/010_Programm/Sudoku/Sudoku.hpp
- C:/Users/benja/Desktop/CSE/PROG/Projekt_Sudoku/010_Programm/Sudoku/Sudoku.cpp

## 3.2 SudokuBacktrack Class Reference

```
#include <SudokuBacktrack.hpp>
```

### Static Public Member Functions

- static bool solveBacktracking (int field[9][9])

  *Die Funktion, die den Backtracking Algorithmus ausführt.*

### Static Private Member Functions

- static bool correctPart (int, int, int, int, int, int field[9][9])

  *Die Funktion correctPart.*
- static bool empty (int field[9][9])

  *Die Funktion empty.*

## 3.2.1 Detailed Description

**Author**

annak

**Date**

7/17/2023

## 3.2.2 Member Function Documentation

### 3.2.2.1 correctPart()

```
bool SudokuBacktrack::correctPart (
            int rowStart,
            int colStart,
            int rowEnd,
            int colEnd,
            int sol,
            int field[9][9] )  [static], [private]
```

Die Funktion correctPart.

Diese Funktion überprüft, ob die ausgewählte Zahl richtig ist. Die Parameter geben eine Teilmatrix an. Innerhalb der Teilmatrix wird überprüft, ob die Zahl bereits vorkommt. Falls ja, ist der Teil nicht korrekt, es wird ¨false¨ zurückgegeben. Ansonsten ¨true¨.

**Parameters**

| | |
|---|---|
| *rowStart* | Anfangszeile |
| *colStart* | Anfangsspalte |
| *rowEnd* | Endzeile |
| *colEnd* | Endspalte |
| *sol* | ausgewählte Zahl für ein Feld |
| *field* | die Matrix, die das Sudoku enthält |

**Returns**

Korrektheit der Teilmatrix

#### 3.2.2.2 empty()

```
bool SudokuBacktrack::empty (
            int field[9][9] )  [static], [private]
```

Die Funktion empty.

Die Funktion durchläuft in zwei verschachtelten Schleifen die Matrix und überprüft, ob es noch ein freies Feld gibt. Falls ja, wird ¨true¨ zurückgegeben. Ansonsten ¨false¨.

**Parameters**

| | |
|---|---|
| *field* | die Matrix, die das Sudoku enthält |

**Returns**

Existenz eines freien Feldes im Sudoku

#### 3.2.2.3 solveBacktracking()

```
bool SudokuBacktrack::solveBacktracking (
            int field[9][9] )  [static]
```

Die Funktion, die den Backtracking Algorithmus ausführt.

Zuerst wird mit der empty Funktion überprüft, ob es noch freie Felder im Sudoku gibt. Falls ja, werden die leeren Felder nacheinander ausgewählt und von 9 absteigend eine Zahl ausgewählt. Mit der correctPart Funktion wird überprüft, ob die Zahl in der Zeile, Spalte und Block korrekt ist. Falls ja, wird sie eingesetzt. Falls nein, wird die nächsten Zahl überprüft. Die Funktion wird rekursiv aufgerufen und durchläuft alle leeren Felder. Wenn alle Felder korrekt ausgefüllt sind, wird die Funktion beendet und ¨true¨ zurückgegeben.

**Parameters**

| | |
|---|---|
| *field* | die Matrix, die das Sudoku enthält |

**Returns**

Angabe, ob Sudoku gelöst ist

The documentation for this class was generated from the following files:

- C:/Users/benja/Desktop/CSE/PROG/Projekt_Sudoku/010_Programm/Sudoku/SudokuBacktrack.hpp
- C:/Users/benja/Desktop/CSE/PROG/Projekt_Sudoku/010_Programm/Sudoku/SudokuBacktrack.cpp

# Chapter 4

# File Documentation

## 4.1 C:/Users/benja/Desktop/CSE/PROG/Projekt_Sudoku/010_↩ Programm/Sudoku/main.cpp File Reference

```
#include <iostream>
#include ¨Sudoku.hpp¨
#include ¨SudokuBacktrack.hpp¨
```

### Functions

- void testRoutine (string csvName)

  *Feld wird auf der Konsole ausgegeben und anschliessend mit Backtracking geloesst. Sofern es richtig geloesst wurde, wird das Feld erneut ausgegeben.*
- int main (int argc, char ∗∗argv)

### 4.1.1 Function Documentation

#### 4.1.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

#### 4.1.1.2 testRoutine()

```
void testRoutine (
            string csvName )
```

Feld wird auf der Konsole ausgegeben und anschliessend mit Backtracking geloesst. Sofern es richtig geloesst wurde, wird das Feld erneut ausgegeben.

## 4.2 C:/Users/benja/Desktop/CSE/PROG/Projekt_Sudoku/010_↩ Programm/Sudoku/Sudoku.cpp File Reference

```
#include <cmath>
#include <iostream>
#include <fstream>
#include <string>
#include ¨Sudoku.hpp¨
#include ¨SudokuBacktrack.hpp¨
```

## 4.3 C:/Users/benja/Desktop/CSE/PROG/Projekt_Sudoku/010_↩ Programm/Sudoku/Sudoku.hpp File Reference

Klasse fuer ein Sudoku Spiel.

```
#include <iostream>
```

### Classes

- class Sudoku

### 4.3.1 Detailed Description

Klasse fuer ein Sudoku Spiel.

Die Sudoku Klasse bildet ein Sudoku Spiel ab. Sie enthält ein Spielfeld und Informationen über dessen Zustand. Das Ausgangsfeld wird beim Erzeugen des Objekts eingelesen. Es sind Methoden zum Ueberpruefen des Feldes implementiert.

## 4.4 Sudoku.hpp

Go to the documentation of this file.

```
00001 #include <iostream>
00002
00003 using namespace std;
00004
00017 class Sudoku{
00018 private:
00019 // VAR
00025     int field[9][9];
00026
00027     // Attribute
00031     bool solved;
00035     bool zeroes;
00039     bool mistake;
00040
00041 public:
00042 // KONSTRUKTOREN / EINGABE
00043     Sudoku(string);
00044     Sudoku(int f[9][9]);
00045
00046 // VERARBEITUNG
00047     void solve();
```

```
00048
00049 // AUSGABE
00050     void printField();
00051
00052 //VALIDIERUNG
00053 private:
00054     bool checkForZeroes();
00055     bool checkForMistakes();
00056     bool checkBlock(int,int,int,int);
00057 public:
00058     bool checkSudoku();
00059 };
```

## 4.5 C:/Users/benja/Desktop/CSE/PROG/Projekt_Sudoku/010_↩ Programm/Sudoku/SudokuBacktrack.cpp File Reference

```
#include ¨SudokuBacktrack.hpp¨
```

## 4.6 C:/Users/benja/Desktop/CSE/PROG/Projekt_Sudoku/010_↩ Programm/Sudoku/SudokuBacktrack.hpp File Reference

Klasse zum Lösen von Sudokus mit Backtracking.

```
#include <iostream>
```

### Classes

- class SudokuBacktrack

### 4.6.1 Detailed Description

Klasse zum Lösen von Sudokus mit Backtracking.

Diese Klasse implementiert in der cpp Datei den Backtracking Algorithmus zum Lösen eines Sudokus. Die Funktionen correctPart, empty und solveBacktracking werden verwendet.

## 4.7 SudokuBacktrack.hpp

Go to the documentation of this file.

```
00001 #include <iostream>
00002
00003 using namespace std;
00004
00015 class SudokuBacktrack{
00016 private:
00017     static bool correctPart(int, int, int, int, int, int field[9][9]);
00018     static bool empty(int field[9][9]);
00019
00020 public:
00021     static bool solveBacktracking(int field[9][9]);
00022 };
```