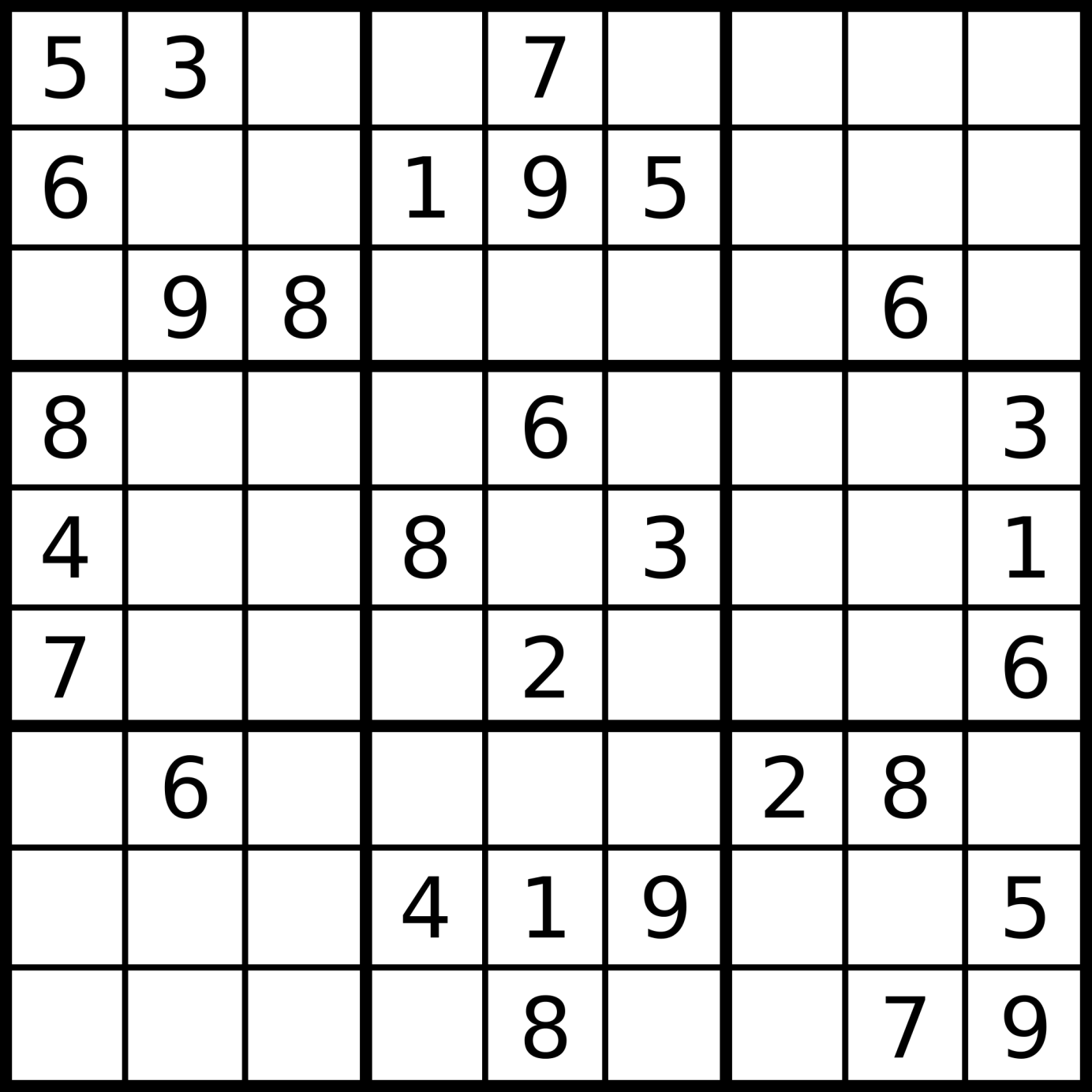
# Deckblatt

**Sudoku**

# 

# Inhaltsverzeichnis

[Deckblatt 1](#_Toc486241712)

[Inhaltsverzeichnis 2](#_Toc486241713)

[Abbildungsverzeichnis 4](#_Toc486241714)

[Impressum 5](#_Toc486241715)

[1. Datenbank 5](#_Toc486241716)

[1.1 accounts 5](#_Toc486241717)

[1.1.1 Create statement 5](#_Toc486241718)

[1.1.2 Aufbau 5](#_Toc486241719)

[1.2 games 5](#_Toc486241720)

[1.2.1 Create statement 5](#_Toc486241721)

[1.2.2 Aufbau 6](#_Toc486241722)

[1.3 sqlite\_sequence 6](#_Toc486241723)

[1.3.1 Create Statement 6](#_Toc486241724)

[1.3.2 Aufbau 6](#_Toc486241725)

[1.4 sudoku\_preset\_fields 6](#_Toc486241726)

[1.4.1 Create Statement 6](#_Toc486241727)

[1.4.2 Aufbau 7](#_Toc486241728)

[1.5 sudoku\_preset\_games 7](#_Toc486241729)

[1.5.1 Create Statement 7](#_Toc486241730)

[1.5.2 Aufbau 7](#_Toc486241731)

[2. Programmstruktur 7](#_Toc486241732)

[2.1 Utils.h 7](#_Toc486241733)

[2.1.1 Definitionen/Inkludierungen 7](#_Toc486241734)

[2.2 Utils.c 8](#_Toc486241735)

[2.2.1 Definitionen/Inkludierungen 8](#_Toc486241736)

[2.2.2 Funktionen 8](#_Toc486241737)

[2.3 globals.h 8](#_Toc486241738)

[2.3.1 Definitionen/Inkludierungen 8](#_Toc486241739)

[2.4 globals.c 8](#_Toc486241740)

[2.4.1 Definitionen/Inkludierungen 8](#_Toc486241741)

[2.4.2 Funktionen 9](#_Toc486241742)

[2.5 highscore.h 9](#_Toc486241743)

[2.5.1 Definitionen/Inkludierungen 9](#_Toc486241744)

[2.6 highscore.c 9](#_Toc486241745)

[2.6.1 Definitionen/Inkludierungen 9](#_Toc486241746)

[2.6.2 Funktionen 9](#_Toc486241747)

[2.7 login.h 11](#_Toc486241748)

[2.7.1 Definitionen/Inkludierungen 11](#_Toc486241749)

[2.8 login.c 11](#_Toc486241750)

[2.8.1 Definitionen/Inkludierungen 11](#_Toc486241751)

[2.8.2 Funktionen 11](#_Toc486241752)

[2.9 sudoku.h 12](#_Toc486241753)

[2.9.1 Definitionen/Inkludierungen 12](#_Toc486241754)

[2.10 sudoku.c 13](#_Toc486241755)

[2.10.1 Definitionen/Inkludierungen 13](#_Toc486241756)

[2.10.2 Funktionen 13](#_Toc486241757)

[2.11 ui.h 17](#_Toc486241758)

[2.11.1 Definitionen/Inkludierungen 17](#_Toc486241759)

[2.12 ui.c 18](#_Toc486241760)

[2.12.1 Definitionen/Inkludierungen 18](#_Toc486241761)

[2.12.2 Funktionen 18](#_Toc486241762)

[2.13 ui\_logic.h 26](#_Toc486241763)

[2.13.1 Definitionen/Inkludierungen 26](#_Toc486241764)

[2.14 ui\_logic.c 26](#_Toc486241765)

[2.14.1 Definitionen/Inkludierungen 26](#_Toc486241766)

[2.14.2 Funktionen 26](#_Toc486241767)

[2.15 registration.h 30](#_Toc486241768)

[2.15.1 Definitionen/Inkludierungen 30](#_Toc486241769)

[2.16 registration.c 30](#_Toc486241770)

[2.16.1 Definitionen/Inkludierungen 30](#_Toc486241771)

[2.16.2 Funktionen 30](#_Toc486241772)

[2.17 constants.h 31](#_Toc486241773)

[2.17.1 Definitionen/Inkludierungen 31](#_Toc486241774)

[3. Struktogramme 32](#_Toc486241775)

[3.1 mainMenu 32](#_Toc486241776)

[3.2 registration 33](#_Toc486241777)

[3.3 testIfUserNameExists 33](#_Toc486241778)

[3.4 insertNewUser 34](#_Toc486241779)

[3.5 handelRegistration 34](#_Toc486241780)

[4. Änderungen gegenüber dem Pflichtenheft 35](#_Toc486241781)

[4.1 Bestenliste 35](#_Toc486241782)

# Abbildungsverzeichnis

[Abbildung [mainMenu]](#_Toc485886230) [32]

[Abbildung [registration] [33]](#_Toc485886231)

[Abbildung [testIfUserNameExists] [33]](#_Toc485886231)

[Abbildung [insertNewUser] [34]](#_Toc485886231)

[Abbildung [handelRegistration] [34]](#_Toc485886231)

# 

# Impressum

Titel: Sudoku Handbuch  
Version: 1.0  
Herausgeber: HHBKTendo Research Center  
Autoren: Simon Marx, Yannik Knops, Joel Kamps, Timo Scheile, Lukas Knapp, Nick Schikora  
Freigabedatum: 26.06.2017  
Firmenanschrift:   
Copyright: ©2017 Herausgeber

# 

# 1. Datenbank

## 1.1 accounts

### 1.1.1 Create statement

CREATE TABLE `accounts` (

`id` INTEGER PRIMARY KEY AUTOINCREMENT,

`username` TEXT NOT NULL UNIQUE,

`passwort` TEXT NOT NULL

)

### 1.1.2 Aufbau

|  |  |
| --- | --- |
| Spalte | Typ |
| ID | Integer |
| Username | Text |
| Passwort | Text |

## 1.2 games

### 1.2.1 Create statement

CREATE TABLE `games` (

`id` INTEGER PRIMARY KEY AUTOINCREMENT,

`user\_id` INTEGER NOT NULL,

`total\_game\_time` INTEGER NOT NULL,

`difficulty` INTEGER NOT NULL,

`punkte` INTEGER NOT NULL DEFAULT 0

)

### 1.2.2 Aufbau

|  |  |
| --- | --- |
| Spalte | Typ |
| ID | Integer |
| User\_ID | Integer |
| Total\_Game\_Time | Integer |
| Difficulty | Integer |
| Punkte | Integer |

## 1.3 sqlite\_sequence

### 1.3.1 Create Statement

CREATE TABLE sqlite\_sequence(name,seq)

### 1.3.2 Aufbau

|  |  |
| --- | --- |
| Spalte | Typ |
| Name | Text |
| Seq | Text |

## 1.4 sudoku\_preset\_fields

### 1.4.1 Create Statement

CREATE TABLE "sudoku\_preset\_fields" (

`id` INTEGER PRIMARY KEY AUTOINCREMENT,

`preset\_id` INTEGER,

`x\_cordinate` INTEGER,

`y\_cordinate` INTEGER,

`value` INTEGER,

`is\_solution` INTEGER

)

### 1.4.2 Aufbau

|  |  |
| --- | --- |
| Spalte | Typ |
| ID | Integer |
| Preset\_ID | Integer |
| X\_cordinate | Integer |
| Y\_cordinate | Integer |
| Value | Integer |
| Is\_Solution | Integer |

## 1.5 sudoku\_preset\_games

### 1.5.1 Create Statement

CREATE TABLE `sudoku\_preset\_games` (

`id` INTEGER,

`difficulty` INTEGER

)

### 1.5.2 Aufbau

|  |  |
| --- | --- |
| Spalte | Typ |
| ID | Integer |
| Difficulty | Integer |

# 2. Programmstruktur

## 2.1 Utils.h

### 2.1.1 Definitionen/Inkludierungen

#ifndef \_\_UTILS\_\_  
#define \_\_UTILS\_\_  
#define DB\_FILE „sodoku\_gruppe\_a.db“  
#endif

|  |
| --- |
| #include <stdio.h> |
| #include <stdlib.h> |
| #include <string.h> |
| #include <Windows.h> |
| #include <conio.h> |
|  |
| #include "sqlite3.h" |

## 2.2 Utils.c

### 2.2.1 Definitionen/Inkludierungen

#include „../inc/Utils.h“

### 2.2.2 Funktionen

/\*

\* =============================================================================

\* initExistingValuesArrays

\* Parameter: existingValues, maxIndex

\* Rückgabewert: /

\* Beschreibung: Füllt das ganze Array mit Nullen

\* ============================================================================

\*/

## 2.3 globals.h

### 2.3.1 Definitionen/Inkludierungen

#ifndef \_\_USER\_ID\_\_  
#define \_\_USER\_ID\_\_  
extern int iUserId;  
#endif

## 2.4 globals.c

### 2.4.1 Definitionen/Inkludierungen

#include "../inc/globals.h"

### 2.4.2 Funktionen

Int iUserId = 0

## 2.5 highscore.h

### 2.5.1 Definitionen/Inkludierungen

|  |
| --- |
| #ifndef \_\_HIGHSCORE\_\_ |
| #define \_\_HIGHSCORE\_\_ |
|  |
| #include "Utils.h" |
| #include "ui.h" |
|  |
| void databaseCallHighscore(char \*sQuery); |
| static int callbackBestenliste(void \*data, int argc, char \*\*argv, |
| char \*\*colName); |
| void showHighscore(); |
| void getHighscoreTable(); |
| void checkIfColIsPoint(char \*const \*argv, char \*const \*colName, int i); |
| void checkIfColIsName(char \*const \*argv, char \*const \*colName, int i); |
|  |
| #endif |

## 2.6 highscore.c

### 2.6.1 Definitionen/Inkludierungen

#include "../inc/highscore.h"

### 2.6.2 Funktionen

/\*

\* =============================================================================

\* getHighscoreTable

\* Parameter: -

\* Rückgabewert: -

\* Beschreibung: Zusammenbau einer Select-Query und aufruf der ShowHighscore

\* und databaseCallHighscore Methode

\* ============================================================================

\*/

/\*

\* =============================================================================

\* databaseCallHighscore

\* Parameter: char \*sQuery

\* Rückgabewert: -

\* Beschreibung: Ausfüren eines DatenbankCalls mittels übergeber Query unter

\* Zuhilfe nahme einer Callback-Funktion

\* ============================================================================

\*/

/\*

\* =============================================================================

\* callbackBestenliste

\* Parameter: void \*data, int argc, char \*\*argv, char \*\*colName

\* Rückgabewert: 0

\* Beschreibung: Callback-Funktion für Bestenlisten Darstellung

\* ============================================================================

\*/

/\*

\* =============================================================================

\* checkIfColIsPoint

\* Parameter: char \*const \*argv, char \*const \*colName, int i

\* Rückgabewert: 0

\* Beschreibung: Prüfung ob Spaltenname "name" ist. Und formatierte Ausgabe.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* checkIfColIsName

\* Parameter: char \*const \*argv, char \*const \*colName, int i

\* Rückgabewert: 0

\* Beschreibung: Prüfung ob Spaltenname "punkte" ist. Und formatierte Ausgabe.

\* ============================================================================

\*/

## 2.7 login.h

### 2.7.1 Definitionen/Inkludierungen

|  |
| --- |
| #ifndef \_\_LOGIN\_\_ |
| #define \_\_LOGIN\_\_ |
| #include "Utils.h" |
| #include "globals.h" |
|  |
| void databaseCallLogin(char \*sQuery); |
| static int callbackLogin(void \*data, |
| int argc, |
| char \*\*argv, |
| char \*\*colName); |
| int loginUser(char \*cName, char \*cPassword); |
| #endif |

## 2.8 login.c

### 2.8.1 Definitionen/Inkludierungen

#include "../inc/login.h"

### 2.8.2 Funktionen

/\*

\* =============================================================================

\* loginUser

\* Parameter: char \*cName, char \*cPassword

\* Rückgabewert: 0

\* Beschreibung: Bauen der Select-Query auf Basis von Übergabeparametern und

\* aufruf der databaseCallLogin-Methode.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* databaseCallLogin

\* Parameter: char \*sQuery

\* Rückgabewert: -

\* Beschreibung: Öffnen der Datenbank und ausführen der übergebenen Query

\* ============================================================================

\*/

/\*

\* =============================================================================

\* callbackLogin

\* Parameter: void \*data, int argc, char \*\*argv, char \*\*colName

\* Rückgabewert: 0

\* Beschreibung: Callback Funktion zur Zuweisung einer Datenbank ID auf

\* globale iUserId

\* ============================================================================

\*/

## 2.9 sudoku.h

### 2.9.1 Definitionen/Inkludierungen

|  |
| --- |
| #ifndef \_\_SUDOKU\_\_ |
| #define \_\_SUDOKU\_\_ |
| #include "Utils.h" |
|  |
| #include "customTypes\sudoku\_field.h" |
| #include "constants.h" |
|  |
| // prototypes |
| int initSudoku(int difficulty, sudoku\_field sudokuFields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS]); |
| int validateSudoku(sudoku\_field sudokuFields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS]); |
| int validateRow(int rowXIndex, int rowYIndex, sudoku\_field sudokuFields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS]); |
| int validateRows(sudoku\_field sudokuFields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS]); |
| int validateField(); |
| #endif |

## 2.10 sudoku.c

### 2.10.1 Definitionen/Inkludierungen

#include "../inc/sudoku.h"

### 2.10.2 Funktionen

|  |
| --- |
| /\* |
| \* =================================================================== |
| \* initSudoku |
| \* Parameter: sudoku\_field |
| \* sudokuFields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS] |
| \* Dieses zweidimensionale Array enthält alle Felder des Sudokus. |
| \* int iWithSolution Gibt an, ob das Spielfeld aufgelöst werden soll |
| \* Rückgabewert: - |
| \* Beschreibung: Die Funktion lädt eines der in der Datenbank gespeicherten |
| \* Spiele mit dem gewünschten Schwierigkeitsgrad und füllt |
| \* die Spieldaten in das Array. |
| \* =================================================================== |
| \*/ |
| /\* |
| \* =================================================================== |
| \* validateSudoku |
| \* Parameter: sudoku\_field |
| \* sudokuFields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS] |
| \* Dieses zweidimensionale Array enthält alle Felder des Sudokus. |
| \* Rückgabewert: int Gibt 1 zurück, wenn das Sudoku komplett valide ist und |
| \* alle Regeln befolgt wurden, 0 wenn nicht. |
| \* Beschreibung: Die Funktion überprüft alle Zeilen und Spalten sowie, ob alle |
| \* Felder gesetzt sind. Sollten alle Zeile und Spalten valide und |
| \* alle Felder ausgefüllt sein, liefert die Funktion SUDOKU\_TRUE |
| \* zurück, sollten nicht alle Felder valide sein, liefert die |
| \* Funktion SUDOKU\_FALSE zurück. |
| \* =================================================================== |
| \*/ |
| /\* |
| \* =================================================================== |
| \* areAllFieldsFilledOut |
| \* Parameter: sudoku\_field |
| \* sudokuFields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS] |
| \* Dieses zweidimensionale Array enthält alle Felder des Sudokus. |
| \* Rückgabewert: int SUDOKU\_TRUE wenn alle Felder ausgefüllt sind/ |
| \* SUDOKU\_FALSE wenn NICHT alle Felder ausgefüllt sind |
| \* Beschreibung: Die Funktion überprüft, ob alle Felder im Sudoku gesetzt sind |
| \* =================================================================== |
| \*/ |
| /\* |
| \* =================================================================== |
| \* validateRow |
| \* Parameter: int iRowXIndex Der X Index, dessen Zeile/Spalte überprüft werden |
| \* soll. |
| \* int iRowYIndex Der Y Index, dessen Zeile/Speile überprüft werden |
| \* soll. |
| \* sudoku\_field |
| \* sudokuFields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS] |
| \* Dieses zweidimensionale Array enthält alle Felder des Sudokus. |
| \* Rückgabewert: int Gibt entweder SUDOKU\_TRUE zurück wenn Zeile und Spalte |
| \* der übergebenen Koordinaten valide sind. Gibt SUDOKU\_FALSE |
| \* zurück, wenn Zeile und Spalte nicht valide sind. |
| \* Beschreibung: Die Funktion überprüft, ob die Zahl an den angegebenen |
| \* Koordinaten in der Zeile und der Spalte bereits existieren und |
| \* gibt den entsprechenden Rückgabewert zurück. |
| \* =================================================================== |
| \*/   |  | | --- | | /\* | | \* ================================================================== | | \* validateRows | | \* Parameter: sudoku\_field | | \* sudokuFields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS] | | \* Dieses zweidimensionale Array enthält alle Felder des Sudokus. | | \* Rückgabewert: int SUDOKU\_TRUE, wenn alle Zeilen und Spalten des Sudokus | | \* valide sind. SUDOKU\_FALSE, wenn nicht alle Zeilen und | | \* Spalten valide sind. | | \* Beschreibung: Die Funktion überprüft alle Zeilen und Spalten des Sudokus | | \* und gibt den entsprechenden Rückgabewert zurück. | | \* ================================================================== | | \*/   |  | | --- | | /\* | | \* ================================================================= | | \* validateField | | \* Parameter: int iTopLeftXIndex Die Y-Position der linken oberen Ecke des | | \* 3x3 Felds. | | \* int iTopLeftYIndex Die X-Position der linken oberen Ecke des | | \* 3x3 Felds. | | \* sudoku\_field | | \* sudokuFields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS] | | \* Dieses zweidimensionale Array enthält alle Felder des Sudokus. | | \* Rückgabewert: int SUDOKU\_TRUE, wenn die Zahl im angegebenen Feld nur einmal | | existiert, SUDOKU\_FALSE, wenn eine Zahl mehrmals existiert | | \* Beschreibung: Die Funktion überprüft, ob eine Zahl mehrmals in dem | | \* angegebenen Feld vorhanden ist und gibt den entsprechenenden | | \* Rückgabewert zurück. | | \* ================================================================= | | \*/ | | |
| /\* |
| \* =================================================================== |
| \* validateFields |
| \* Parameter: sudoku\_field |
| \* sudokuFields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS] |
| \* Dieses zweidimensionale Array enthält alle Felder des Sudokus. |
| \* Rückgabewert: int Gibt SUDOKU\_TRUE zurück, wenn alle 3x3 valide sind. |
| \* SUDOKU\_FALSE, wenn nicht. |
| \* Beschreibung: Die Funktion überprüft alle 3x3 Felder. |
| \* =================================================================== |
| \*/   |  | | --- | | /\* | | \* ================================================================== | | \* solveSudoku | | \* Parameter: int xPos | | \* int yPos | | \* sudokuFields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS] | | \* Dieses zweidimensionale Array enthält alle Felder des Sudokus. | | \* int iHasError Gibt an, ob das nächste Feld einen Error hat | | \* Rückgabewert: - | | \* Beschreibung: Die Funktion löst das Sudoku | | \* ================================================================== | | \*/   |  | | --- | | /\* | | \* ============================================================================= | | \* solveSudokuTemp | | \* Parameter: int xPos | | \* int yPos | | \* sudokuFields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS] | | \* Dieses zweidimensionale Array enthält alle Felder des Sudokus. | | \* int iHasError Gibt an, ob das nächste Feld einen Error hat | | \* Rückgabewert: Gibt 1 zurück, wenn das Sudoku gelöst werde konnte. | | \* Gibt 0 zurück, wenn das SUdoku nicht gelöst werden konnte. | | \* Beschreibung: Die Funktion löst das Sudoku \*\*\* NICHT FERTIG\*\*\* | | \* Die Zeit hat nicht gereicht, den Algorithmus richtig zu | | \* implementieren. | | \* ============================================================================ | | \*/ | | |

## 2.11 ui.h

### 2.11.1 Definitionen/Inkludierungen

|  |
| --- |
| #ifndef \_\_UI\_\_ |
| #define \_\_UI\_\_ |
|  |
| #include "Utils.h" |
| #include "constants.h" |
| #include "customTypes\sudoku\_field.h" |
|  |
| void center(char cMessage[]); |
| void bigWhiteSpace(); |
| void whiteSpace(); |
| void quadWhiteSpace(); |
| void hexaWhiteSpace(); |
| void lineBreaks(); |
| void printErrorMessage(char \*cError); |
| void showStartScreen(int iSelector); |
| void printLogin(); |
| void printRegistration(); |
| void printInputUsername(); |
| void printInputPassword(); |
| void printInputPasswordRepeat(); |
| void printSuccessMessage(char\* cUsername); |
| void showDifficulty(int iSelector); |
| void showLoggedInStartScreen(int iSelector); |
| void printFieldHorizontal(); |
| void printField(sudoku\_field sudoku\_fields[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS]); |
| void showHighscore(); |
| void printSelector(int iPosition, int iSelector); |
| void showIntro(); |
| #endif |

## 2.12 ui.c

### 2.12.1 Definitionen/Inkludierungen

|  |
| --- |
| #include <stdio.h> |
| #include <stdlib.h> |
| #include <Windows.h> |
|  |
| #include "../inc/ui.h" |

### 2.12.2 Funktionen

/\*

\* =============================================================================

\* printSudokuRules

\* Parameter: /

\* Rückgabewert: /

\* Beschreibung: Schreibt die Sudokuregln in das Konsolenfenster. Die Regeln

\* die geschrieben werden lauten: "In jeder Zeile duerfen die Ziffern von

\* 1 bis 9 nur einmal vorkommen", "In jeder Spalte duerfen die Ziffern von

\* 1 bis 9 nur einmal vorkommen" und "In jedem Block duerfen die Ziffern von

\* 1 bis 9 nur einmal vorkommen".

\* ============================================================================

\*/

/\*

\* =============================================================================

\* center

\* Parameter: char message Array

\* Rückgabewert: /

\* Beschreibung: Zentriert die Nachricht und gibt sie im Konsolenfenster aus.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* whiteSpace

\* Parameter: /

\* Rückgabewert: /

\* Beschreibung: Macht einen Tabstop.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* bigWhiteSpace

\* Parameter: /

\* Rückgabewert: /

\* Beschreibung: Macht insgesamt neun Tabstopps.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* quadWhiteSpace

\* Parameter: /

\* Rückgabewert: /

\* Beschreibung: Macht vier Tabstops.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* hexaWhiteSpace

\* Parameter: /

\* Rückgabewert: /

\* Beschreibung: Macht sechs Tabstops.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* lineBreaks

\* Parameter: /

\* Rückgabewert: /

\* Beschreibung: Die Methode lässt den Cursor 2 Zeilen nach unten springen.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* printErrorMessage

\* Parameter: char cError

\* Rückgabewert: /

\* Beschreibung: Die methode gibt eine Fehlermeldung aus. Sie fängt an mit

\* "Es ist ein Fehler aufgetreten:" und endet mit dem Fehler der als String

\*übergeben wird.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* showStartScreen

\* Parameter: int iSelector

\* Rückgabewert: /

\* Beschreibung: Zeigt den Startbildschirm an, wenn man noch nicht Eingeloggt

\* ist. Er zeigt die folgenden Menuepunkte: "Schnelles Spiel", "Login",

\* "Registrieren" und "Beenden".

\* ============================================================================

\*/

/\*

\* =============================================================================

\* printLogin

\* Parameter: /

\* Rückgabewert: /

\* Beschreibung: Leert das Fenster und schreibt das Wort "Login"

\* in die Konsole.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* printRegistration

\* Parameter: /

\* Rückgabewert: /

\* Beschreibung: Leert das Fenster und schreibt das Wort "Registrierung"

\* in die Konsole.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* printInputUsername

\* Parameter: /

\* Rückgabewert: /

\* Beschreibung: Schreibt zentriert "Bitte geben Sie Ihren Usernamen ein:"

\* in die Konsole.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* printInputPassword

\* Parameter: /

\* Rückgabewert: /

\* Beschreibung: Schreibt zentriert "Bitte geben Sie Ihr Passwort ein:"

\* in die Konsole.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* printInputPasswordRepeat

\* Parameter: /

\* Rückgabewert: /

\* Beschreibung: Schreibt zentriert "Bitte wiederholen Sie das Passwort:"

\* in die Konsole.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* printSuccessMessage

\* Parameter: char Zeiger cUsername

\* Rückgabewert: /

\* Beschreibung: Zeigt an, dass die Registrierung geklappt hat.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* showDifficulty

\* Parameter: int iSelector

\* Rückgabewert: /

\* Beschreibung: Zeigt die drei verschieden Schwierigkeitsstufen "Leicht",

\* "Mittel" und "Schwer" an.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* showLoggedInStartScreen

\* Parameter: int iSelector

\* Rückgabewert: /

\* Beschreibung: Zeigt den Bildschirm für eingeloggte User an mit den Menue-

\* punkten "Spielen", "Bestenliste", "Regeln", "Logout" und "Beenden".

\* ============================================================================

\*/

/\*

\* =============================================================================

\* printFieldHorizontal

\* Parameter: /

\* Rückgabewert: /

\* Beschreibung: Zeichnet die horizontale Linie des Sudohufeldes.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* showPauseMenu

\* Parameter: int iSelector

\* Rückgabewert: /

\* Beschreibung: Zeigt das Pausenmenue an mit den Optionen "Zurueck zum Spiel",

\* "Zeige Loesung" und "Spiel Beenden".

\* ============================================================================

\*/

/\*

\* =============================================================================

\* printField

\* Parameter: sudoku\_field sudoku\_fields[SUDOKU\_FIELDS\_X\_AXIS]

\* [SUDOKU\_FIELDS\_Y\_AXIS]

\* Rückgabewert: /

\* Beschreibung: Zeichnet das komplette Sudokufeld und zeigt in manchen Feldern

\* Ziffern an. Unter dem Feld wird ein kleines Menue angezeigt.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* showHighscore

\* Parameter: /

\* Rückgabewert: /

\* Beschreibung: Zeigt die besten Spieler an mit dem Namen und

\* den erzielten Punkten.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* printSelector

\* Parameter: int iPosition, int iSelector

\* Rückgabewert: /

\* Beschreibung: Zeigt einen Pfeil neben dem ausgewählten Menuepunkt an.

\* ============================================================================

\*/

## 2.13 ui\_logic.h

### 2.13.1 Definitionen/Inkludierungen

|  |
| --- |
| #ifndef \_\_UI\_LOGIC\_\_ |
| #define \_\_UI\_LOGIC\_\_ |
|  |
| #include "Utils.h" |
| #include "globals.h" |
| #include "ui.h" |
| #include "sudoku.h" |
| #include "login.h" |
|  |
| int mainMenu(); |
| int loggedInMenu(); |
| int difficulty(); |
| int sudokuControls(); |
|  |
| void readUsername(char \*cUsername); |
| void readPassword(char \*cPassword); |
| void handleLogin(); |
| void handleRegistration(); |
| void getHighscore(int user\_id); |
| void help(int xPosition, int yPosition, sudoku\_field field[SUDOKU\_FIELDS\_X\_AXIS][SUDOKU\_FIELDS\_Y\_AXIS]); |
| int pauseMenuHandler(); |
|  |
| void navigation(char cKeyPressed[]); |
| #endif |

## 2.14 ui\_logic.c

### 2.14.1 Definitionen/Inkludierungen

|  |
| --- |
| #include "../inc/ui\_logic.h" |
| #include "../inc/sudoku.h" |

### 2.14.2 Funktionen

|  |
| --- |
| /\* |
| \* =================================================================== |
| \* \*\*\*\*help()\*\*\*\* |
| \* Parameter: int xPosition, int yPosition, sudoku\_field field |
| \* Rückgabewert: - |
| \* Beschreibung: Setzt den Cursor auf die Position unter das Sudokufeld |
| \* Schreibt den text "moegliche loesungen": |
| \* Setzt den Cursor dann unter diesen Text und ruft die |
| \* Funktion auf, die die möglichen nummern für das |
| \* aktuelle Feld errechnet |
| \* =================================================================== |
| \*/ |
| /\* |
| \* =================================================================== |
| \* \*\*\*\*pauseMenuHandler()\*\*\*\* |
| \* Parameter: - |
| \* Rückgabewert: int iChoice |
| \* Beschreibung: Die Funktion behandelt die Benutzereingaben im Pausemenü |
| \* dazu wird erst die Funktion für die Darstellung aufgerufen |
| \* und anschließend auf eine Benutzereingabe gewartet |
| \* =================================================================== |
| \*/ |
| /\* |
| \* =================================================================== |
| \* \*\*\*\*sudokuControls()\*\*\*\* |
| \* Parameter: - |
| \* Rückgabewert: - |
| \* Beschreibung: Die Funktion ist für die gesamte Sudokusteuerung zuständig. |
| \* Benutzereingaben werden behandelt und der Cursor auf dem |
| \* Spielfeld bewegt. Werte werden in das Sudokufield-array |
| \* geschrieben und nach jeder nutzereingabe wird das |
| \* Spiel validiert. |
| \* =================================================================== |
| \*/ |
| /\* |
| \* =================================================================== |
| \* \*\*\*\*mainMenu()\*\*\*\* |
| \* Parameter: - |
| \* Rückgabewert: - |
| \* Beschreibung: Die Funktion überprüft Nutzereingaben, setzt den Cursor, |
| \* ruft die Funktion auf, die das Hauptmenü darstellt |
| \* und ruft beim drücken der Enter-Taste die entsprechende |
| \* Funktion auf. |
| \* =================================================================== |
| \*/ |
| /\* |
| \* =================================================================== |
| \* \*\*\*\*mainMenu()\*\*\*\* |
| \* Parameter: - |
| \* Rückgabewert: - |
| \* Beschreibung: Die Funktion überprüft Nutzereingaben, setzt den Cursor, |
| \* ruft die Funktion auf, die das Hauptmenü für |
| \* eingeloggte Benutzer darstellt |
| \* und ruft beim drücken der Enter-Taste die entsprechende |
| \* Funktion auf. |
| \* =================================================================== |
| \*/ |
| /\* |
| \* =================================================================== |
| \* \*\*\*\*difficulty()\*\*\*\* |
| \* Parameter: - |
| \* Rückgabewert: int iSelector |
| \* Beschreibung: Die Funktion überprüft Nutzereingaben, setzt den Cursor, |
| \* ruft die Funktion auf, die die Auswahl der Schwierigkeit |
| \* darstellt. Die Funktion gibt dann einen int zurück, der |
| \* einen Schwierigkeitsgrad darstellt. |
| \* =================================================================== |
| \*/ |

/\*

\* =============================================================================

\* \*\*\*\*readUsername(char \*cUsername)\*\*\*\*

\* Parameter: char \*cUsername

\* RÃ¼ckgabewert: -

\* Beschreibung: Liest die Konsoleneingabe aus und speichert sie im

char-Array cUsername

\* ============================================================================

\*/

/\*

\* =============================================================================

\* \*\*\*\*readPassword(char \*cPassword)\*\*\*\*

\* Parameter: char \*cPassword

\* RÃ¼ckgabewert:

\* Beschreibung: Liest die Konsoleneingabe aus und speichert sie im

char-Array cPassword

\* ============================================================================

\*/

/\*

\* =============================================================================

\* \*\*\*\*handleLogin()\*\*\*\*

\* Parameter: -

\* Rückgabewert: -

\* Beschreibung: Geht durch die einzelnen Aufgaben beim Einloggen.

Sowohl das Aufrufen der Konsolenausgaben, das Einlesen der

Daten vom Nutzer, sowie die Datenbankabfrage und schlieÃŸt

dies bei Erfolg mit einer neuen

Konsolenausgabe (showLoggedInStartScreen) ab.

\* ============================================================================

\*/

/\*

\* =============================================================================

\* \*\*\*\*handleRegistration()\*\*\*\*

\* Parameter: -

\* Rückgabewert: -

\* Beschreibung: Geht durch die einzelnen Aufgaben beim Registrieren.

Sowohl das Aufrufen der Konsolenausgaben, das Einlesen der

Daten vom Nutzer, sowie die Datenbankabfrage und schlieÃŸt

dies bei Erfolg mit einer neuen

Konsolenausgabe (showStartScreen) ab.

\* ============================================================================

\*/

## 2.15 registration.h

### 2.15.1 Definitionen/Inkludierungen

|  |
| --- |
| #include "Utils.h" |
|  |
| int testIfUserNameExists(char \*cName); |
| void insertNewUser(char \*cName, char \*cPasswort); |

## 2.16 registration.c

### 2.16.1 Definitionen/Inkludierungen

#include „../inc/registration.h“

### 2.16.2 Funktionen

/\*

\* =============================================================================

\* insertNewUser

\* Parameter: char \*cName, char \*cPasswort

\* Rückgabewert: -

\* Beschreibung: Einfügen von Username und Passwort in Datenbank

\* ============================================================================

\*/

/\*

\* =============================================================================

\* testIfUserNameExists

\* Parameter: char \*cName

\* Rückgabewert: int iDoesUserExist

\* Beschreibung: Testen ob das übergebene char-Array in der Datenbank

\* vorhanden ist. iDoesUserExist dient als pseudo-boolean für denn fall,

\* dass cName vorhanden ist.

\* ============================================================================

\*/

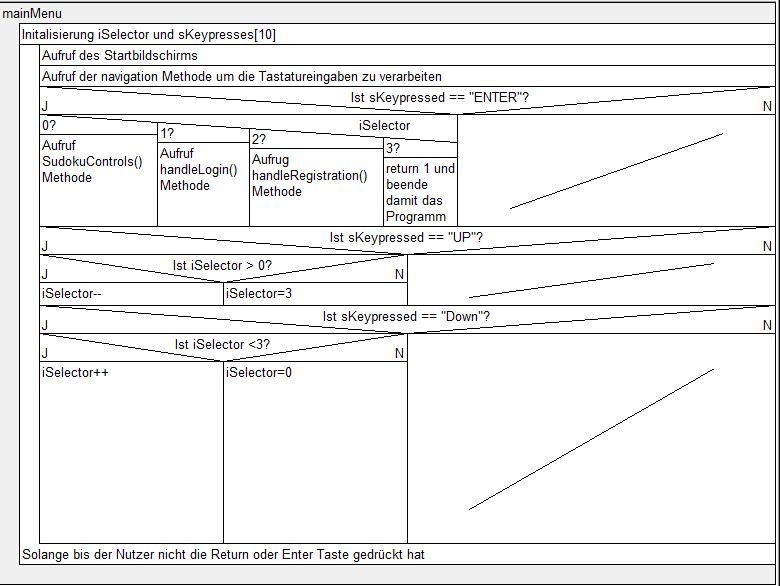
## 2.17 constants.h

### 2.17.1 Definitionen/Inkludierungen

|  |
| --- |
| #ifndef SUDOKU\_FIELDS\_X\_AXIS |
| #define SUDOKU\_FIELDS\_X\_AXIS 9 |
| #endif |
|  |
| #ifndef SUDOKU\_FIELDS\_Y\_AXIS |
| #define SUDOKU\_FIELDS\_Y\_AXIS 9 |
| #endif |

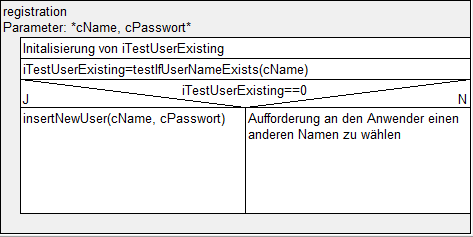
# 3. Struktogramme

## 3.1 mainMenu



**Abb. 1**

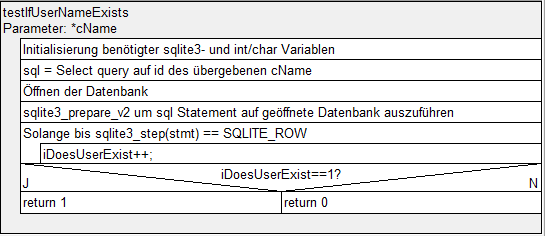
## 3.2 registration



**Abb. 2**

Ruft die Methode testIfUserNameExists auf. Wenn diese 0 zurückgibt wird die Methode insertNewUser aufgerufen.

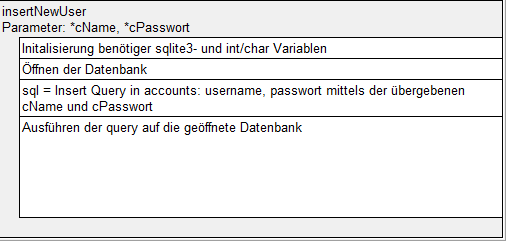
## 3.3 testIfUserNameExists



**Abb. 3**

Guckt in der Datenbank ob der Username bereits vorhanden ist.

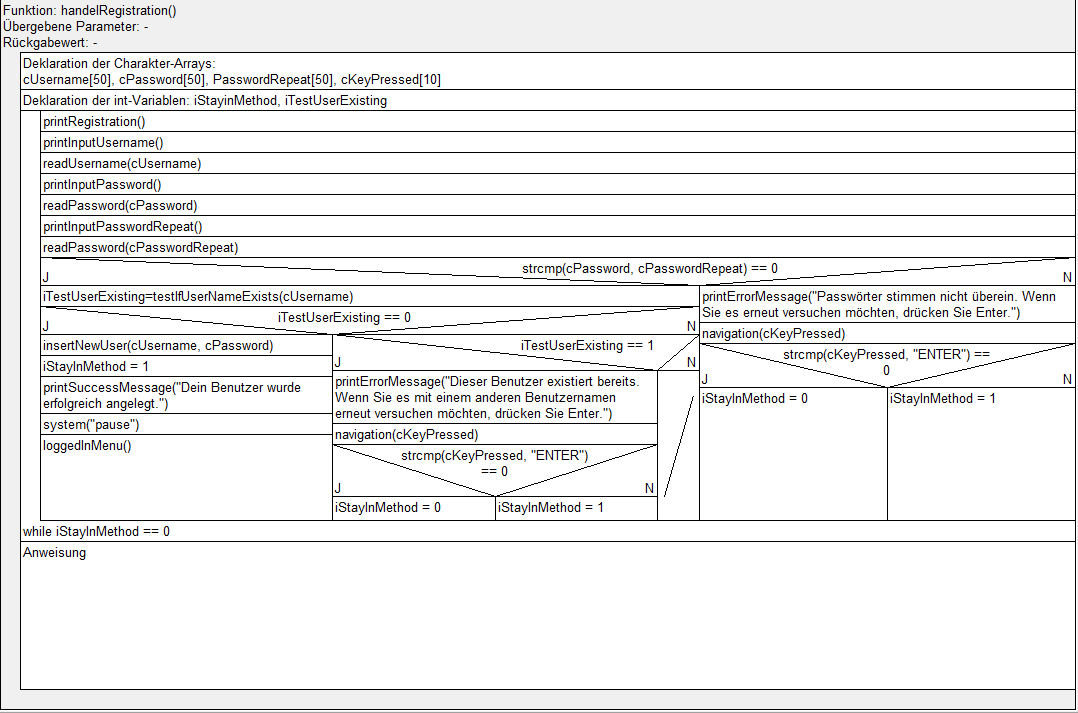
## 3.4 insertNewUser



**Abb. 4**

Fügt den Nutzer in die Datenbank ein.

## 3.5 handelRegistration



**Abb. 5**

# 4. Änderungen gegenüber dem Pflichtenheft

## 4.1 Bestenliste

Es wurde aus Zeitgründen auf eine Ausgabe der eigenen Position in der Bestenliste verzichtet. Ebenso wie eine Darstellung der Platzierung.  
Eine Punkteerrechnung, zur Platzierung, welche aus Schwierigkeitsgrad und gebrauchter Zeit errechnet werden sollte, konnte zeitlich ebenfalls nicht umgesetzt werden, womit eine Sortierung der Platzierung letztlich nicht möglich sein wird, auch wenn eine Spalte für die Punkte in der Datenbank existiert.