Deckblatt

Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Inhaltsverzeichnis

Abbildungsverzeichnis

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:	

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

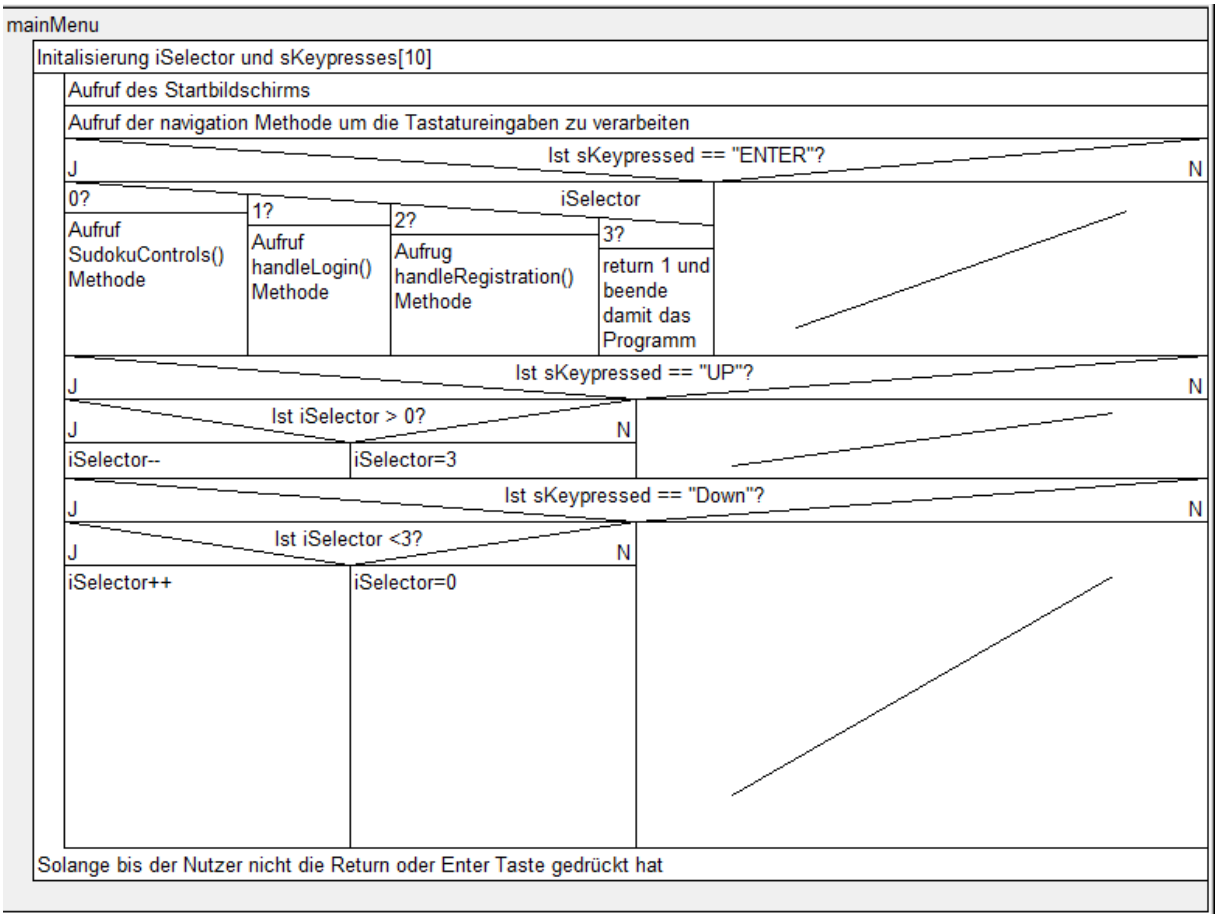

Abb. 1

3.2 registration

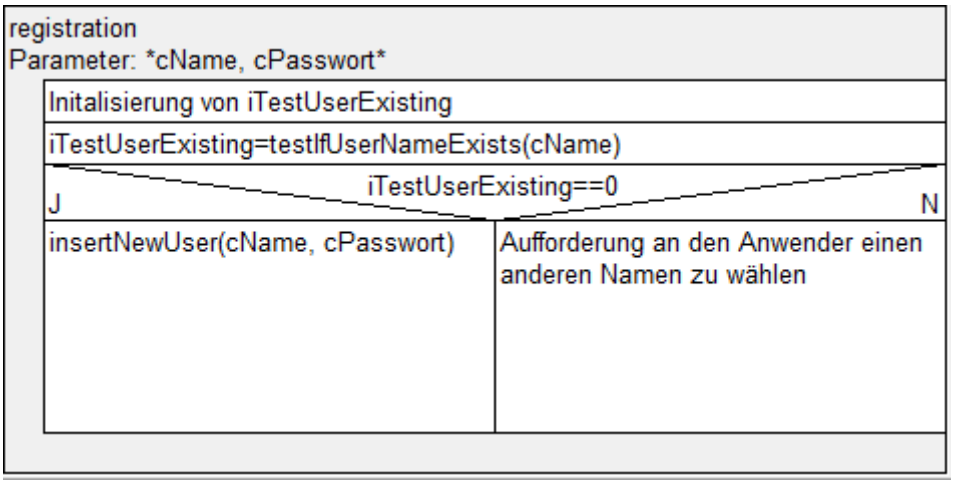

Abb. 2

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

3.3 testIfUserNameExists

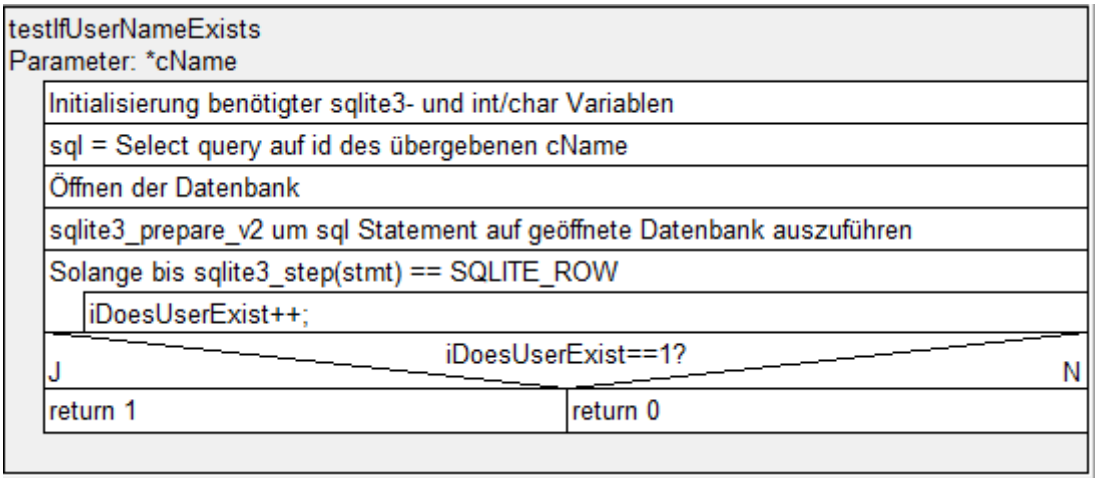

Abb. 3

Guckt in der Datenbank ob der Username bereits vorhanden ist.

3.4 insertNewUser

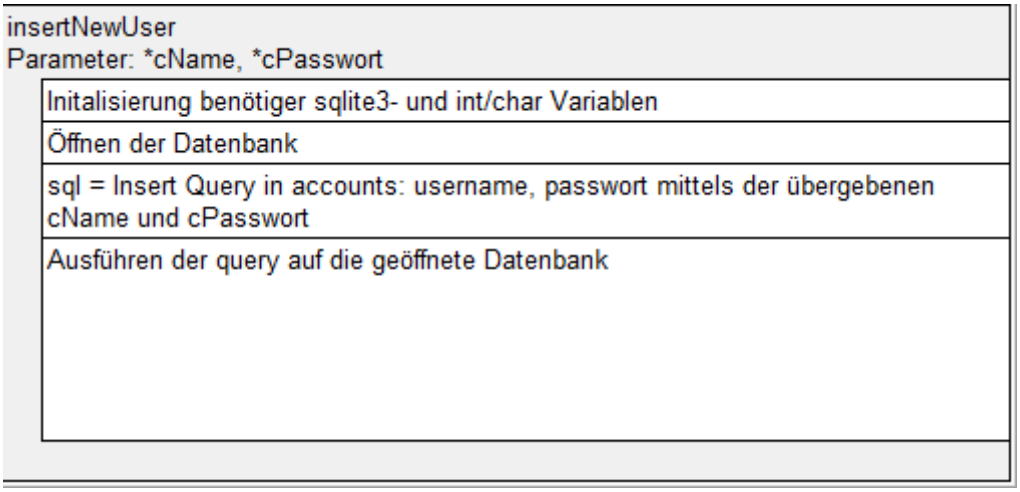

Abb. 4

Fügt den Nutzer in die Datenbank ein.

3.5 handelRegistration

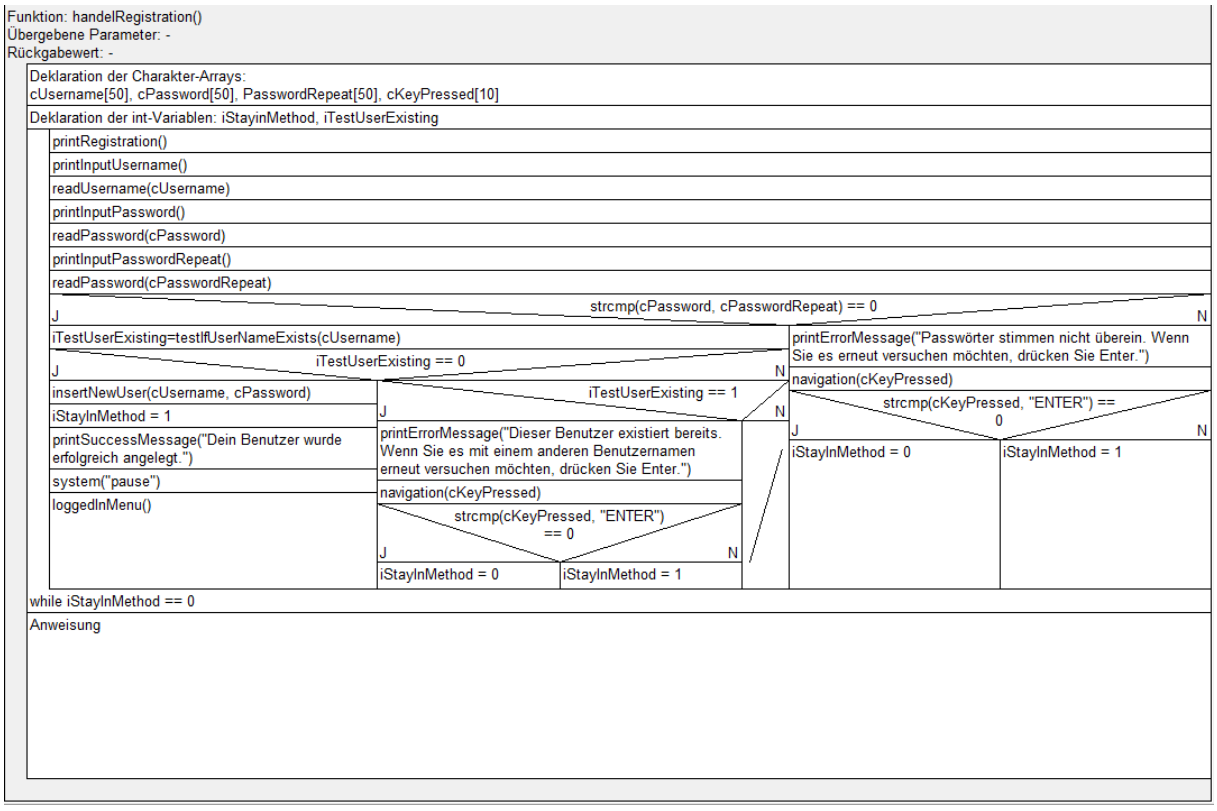

Abb. 5

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.