2018

mentation

Jan Peter Hamm, Maksim Scheierman, Lena Willenbrock

10.04.2018

Inhalt

1 Ausgangssituation

1.1 Projektziele und Teilaufgaben

Das Ziel dieses Projektes ist es, ein fertiges und fehlerfreies Sudoku-Spiel zu erstellen. Der Kunde soll später fehlerfrei damit spielen können und das Sudokufeld fehlerfrei ausfüllen können.

1.2 Kundenanforderungen

Die folgenden Anforderungen sollen erfüllt werden. Dabei ist zwischen den Pflichtanforderungen (gekennzeichnet mit „Muss“) und Erweiterungen (gekennzeichnet mit „Kann“) für das Programm zu unterscheiden:

- Anzeige des Gitters (9x9) -> Muss
  Programm zeigt auf dem Bildschirm ein Gitter aus 9x9 Feldern an, die mit den passenden Zahlen gefüllt werden
- Auswahl des aktuellen Feldes auf dem Spielfeld -> Muss
  Spieler kann auswählen, welches Feld im Gitter bearbeitet werden soll
- Eingabe bzw. Veränderung des Wertes eines Feldes -> Muss
  Spieler kann den Inhalt des aktuellen Feldes im Gitter bearbeiten. Ein Feld darf entweder leer sein oder muss eine Zahl zwischen 1 und 9 enthalten.
- Sichere Verarbeitung der Benutzereingaben -> Muss
  Programm reagiert nur auf gültige Eingaben des Benutzers.
- Auswahl eines zu spielenden Sudoku -> Muss
  Spieler kann ein zu spielendes Sudoku auswählen. Eine Auswahl von Sudoku verschiedener Schwierigkeit kann fest vorgegeben werden. Alternativ können neue Sudoku automatisch erzeugt oder aus einer Datei eingelesen werden (siehe unten).
- Erkennung einer korrekten Lösung des Sudoku -> Muss
  Programm erkennt automatisch oder nach Aufforderung ein korrekt geöstes Sudoku.
- Zeitmessung während des Spiels -> Muss
  Programm zeigt die bereits vergangene Zeit zur Lösung eines Sudoku an. Die Zeit wird entweder am Ende des Spiels, nach jedem Zug oder laufend angezeigt.
- Automatische Erstellung neuer Sudoku -> Kann
  Programm generiert automatisch neue (lösbare) Sudokurätsel verschiedener Schwierigkeit.

- Einlesen eines Sudoku aus einer Datei -> Kann
  Programm liest ein neues Sudoku aus einer Datei ein.
- Abspeichern und Laden eines Spielstandes -> Kann
  Spieler kann den aktuelle Spielstand in einer Datei abspeichern und zu einem späteren Zeitpunkt wieder laden, um das Spiel fortzusetzen.
- Lösungshinweise anzeigen -> Kann
  Programm gibt auf Nachfrage einen gültigen Tipp zur Lösung des Rätsels.

1.3 Projektumfeld

Neben den Anforderungen haben wir außerdem noch einige Rahmenbedingungen bekommen.

Zum einen soll das Programm durch Funktionen und Dateien strukturiert werden.

Alle Dateien, Funktionen und wichtige Quellcodeabschnitte sollen kommentiert werden.

Das Programm soll in der Programmiersprache C mit Hilfe des Programms „CodeBlocks“ und dem damit einhergehenden „MinGW“- Compiler programmiert werden.

Es sollen keine Compiler- oder Linker- Warnungen bzw. –Fehler bei Abgabe vorhanden sein.

Die Benutzung von Bibliotheken ist Erlaubt.

Wenn man einen kleinen Codeabschnitt aus dem Internet zieht, muss die Quelle hinzugefügt werden.

Wir haben uns dazu entschlossen, einen Programmablaufplan zur grafischen Darstellung des Programmes zu erstellen. Außerdem wollen wir die Quellcodeverwaltung mittels Github erleichtern.

Das Abgabedatum ist der 22.05.2018.

1.4 Prozessschnittstellen

Als Ansprechpartner steht uns Herr Wichmann zur Verfügung.

2 Ressourcen und Ablaufplanung

2.1 Projektmitglieder

Unsere Projektgruppe besteht aus Jan Peter Hamm, Maksim Scheierman und Lena Willenbrock.

2.2 Aufgabenverteilung

Wir haben uns darauf geeinigt, dass Jan Peter Hamm und Maksim Scheierman den größeren Teil der Programmierung vornehmen. Lena Willenbrock wird zeitgleich an der Dokumentation arbeiten und ggf. bei der Programmierung unterstützen. Wir haben uns für diese Aufgabenverteilung entschieden, da es in unseren Augen am effektivsten erschien. Jeder kann so selbstständig arbeiten.

Wir haben uns in diesem Zuge für eine Quellcodeverwaltung mit GitHub entschieden, da man dort nach Einarbeitung relativ einfach den Quellcode Verwalten kann. Somit kann jeder auf die aktuelle Version des Quellcodes zugreifen.

3 Durchführung und Auftragsbearbeitung

3.1 Programmablaufplan

Wir haben uns entschieden nur die wichtigsten Kernfunktionen als PAP darzustellen.

Dazu haben wir das Programm PapDesigner verwendet. Die Abbildungen der Paps können ebenfalls mit dem genannten Programm geöffnet werden, die Dateien befinden sich im Dokumentationsordner „Paps“ in der Projektmappe.

Unsere 6 wichtigsten Funktionen:

1. Speichern,
2. Laden des Sudokus
3. Lösen des Sudokus
4. Überprüfung der Lösung
5. Spielloop
6. Chooserloop

1.Die Speicher-Funktion,

int saveGame(SF gameField[9][9], int passedTimeInSeconds):

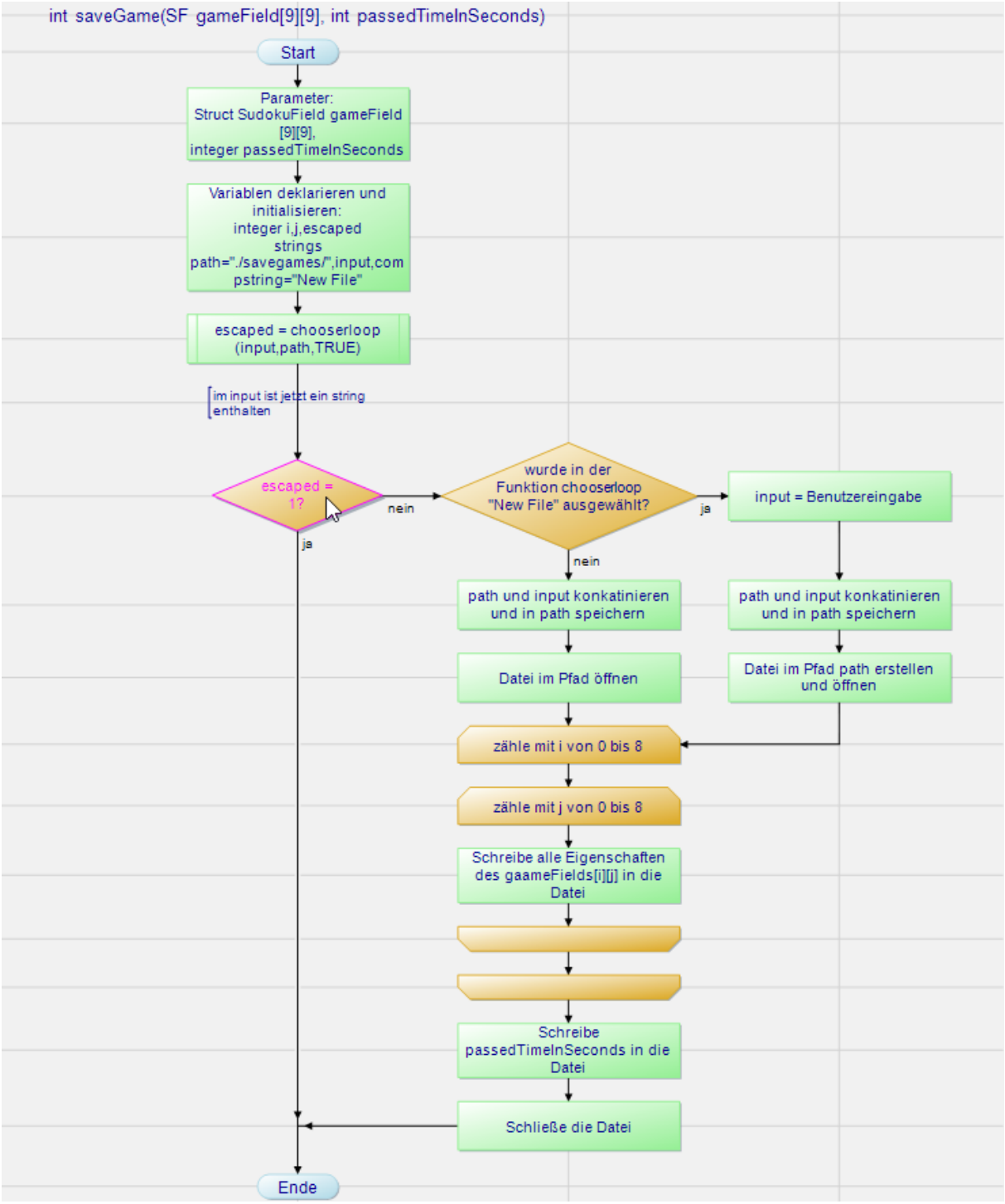

Abbildung 1

2.Die Lade-Funktion int loadGameFromFile():

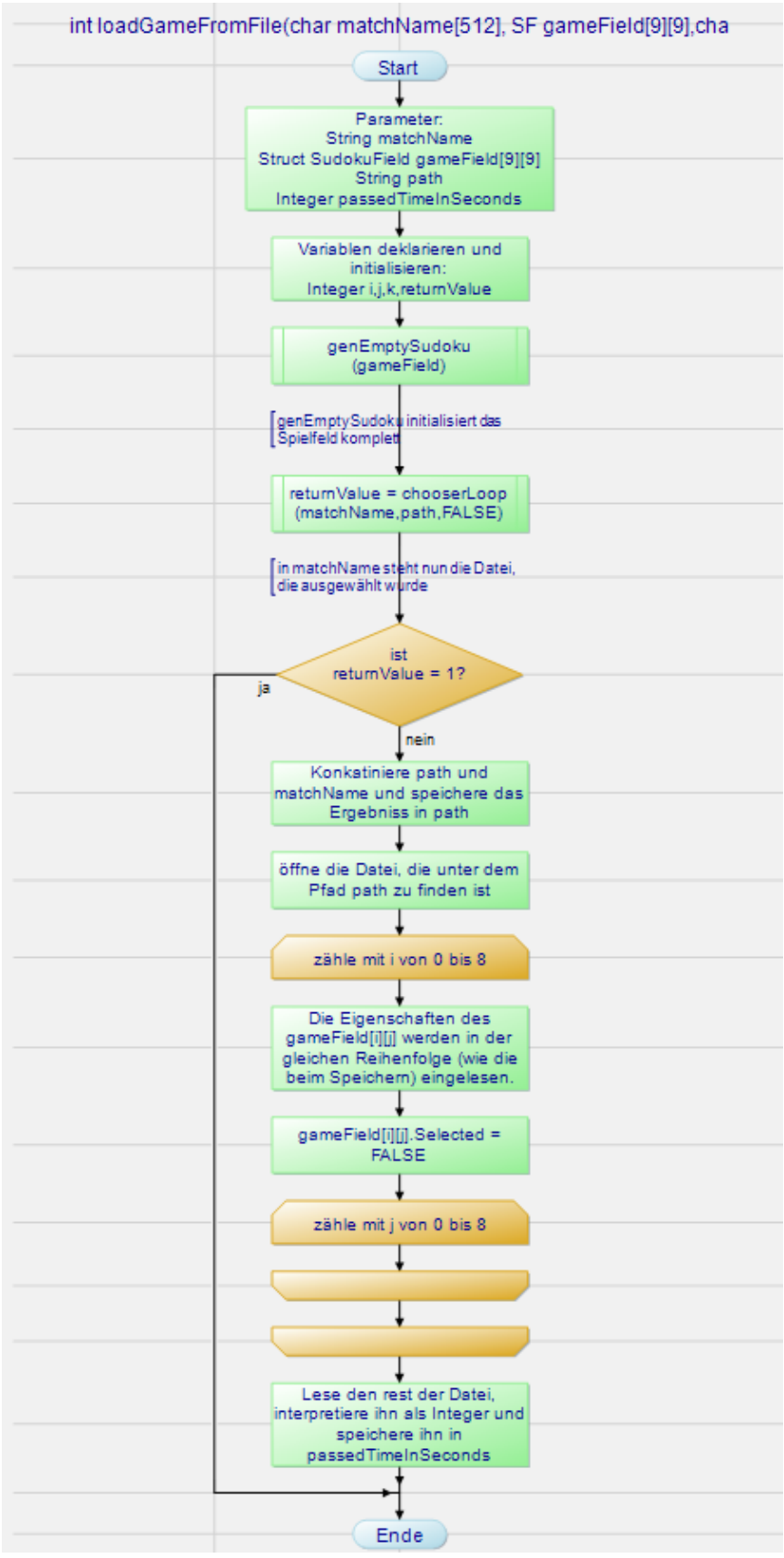

Abbildung 2

3.Die Funktion int sodokuSolver(SF NewMatchField[9][9]) sorgt dafür, dass das ein beliebiges unfertiges Sudoku gelöst wird.(Unterfunktionen als PAP in der Datei)

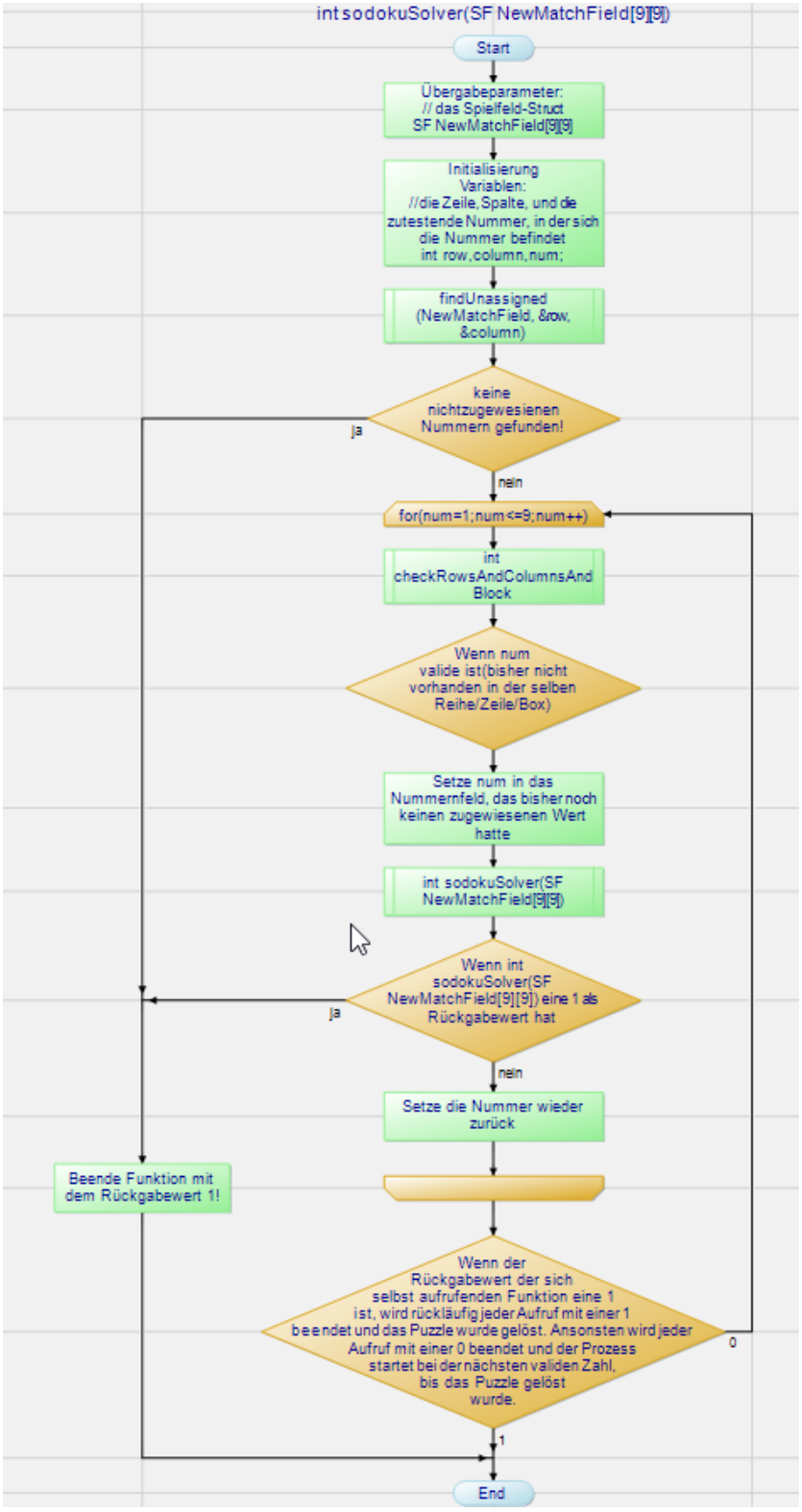

Abbildung 3

4,Die Funktion checkIFSolved, überprüft ob das Sudoku gelöst ist.

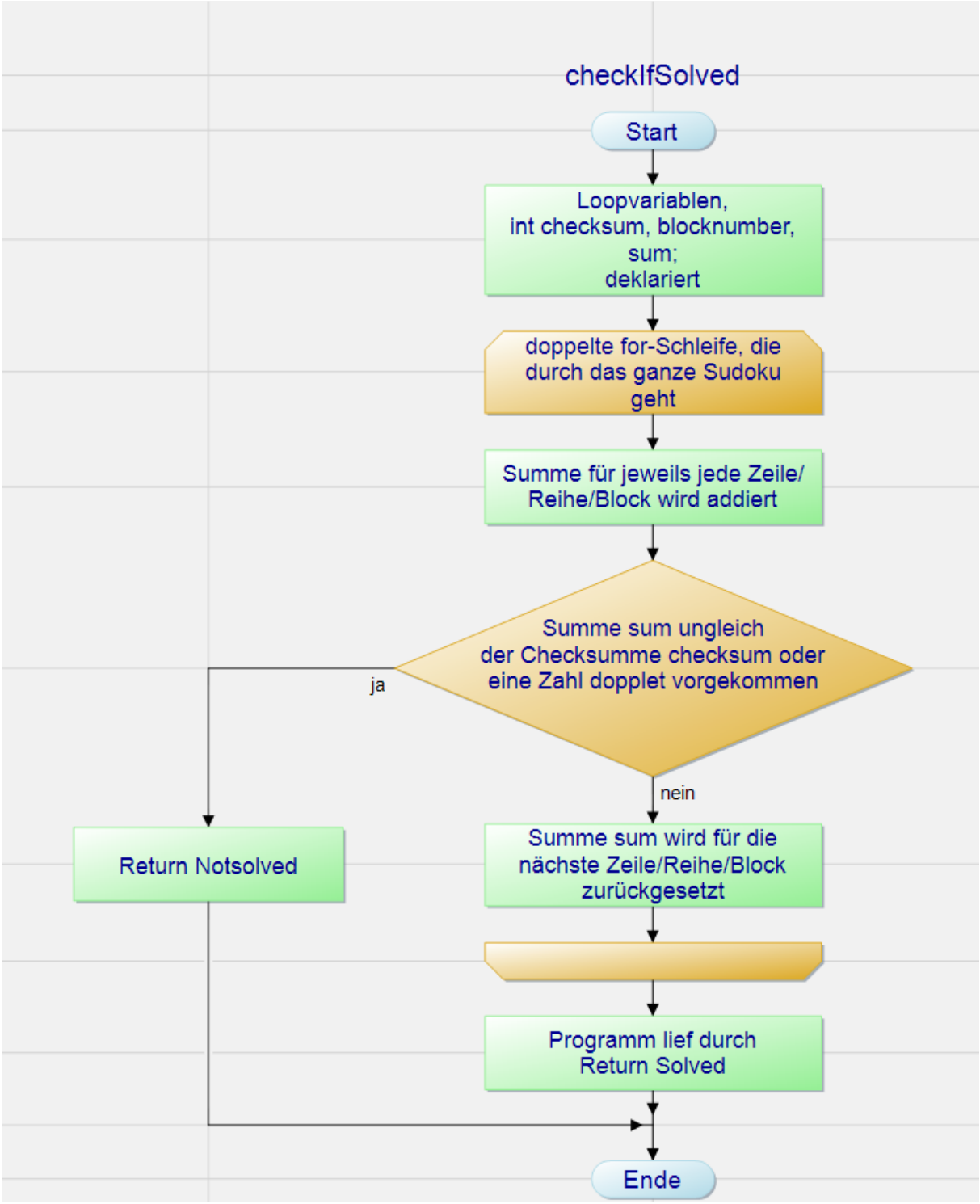

Abbildung 4

5.Die Funktion gameLoop() ist die Funktion die dafür sorgt, dass der Benutzer im Spiel bleibt und abhängig von seiner Auswahl sich der nächste Loop öffnet.

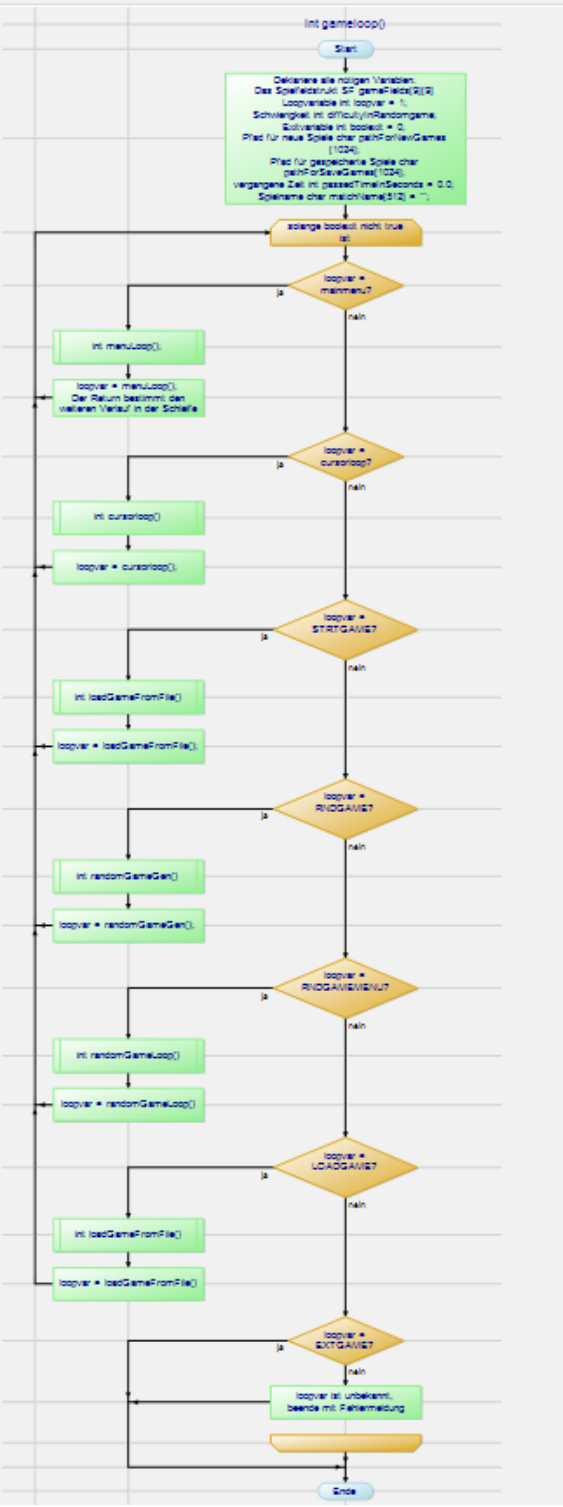

Abbildung 5

6.Die ChooserLoop-Funktion, die dazu dient das Eingaben über Tastatur eingelesen und bearbeitet werden.

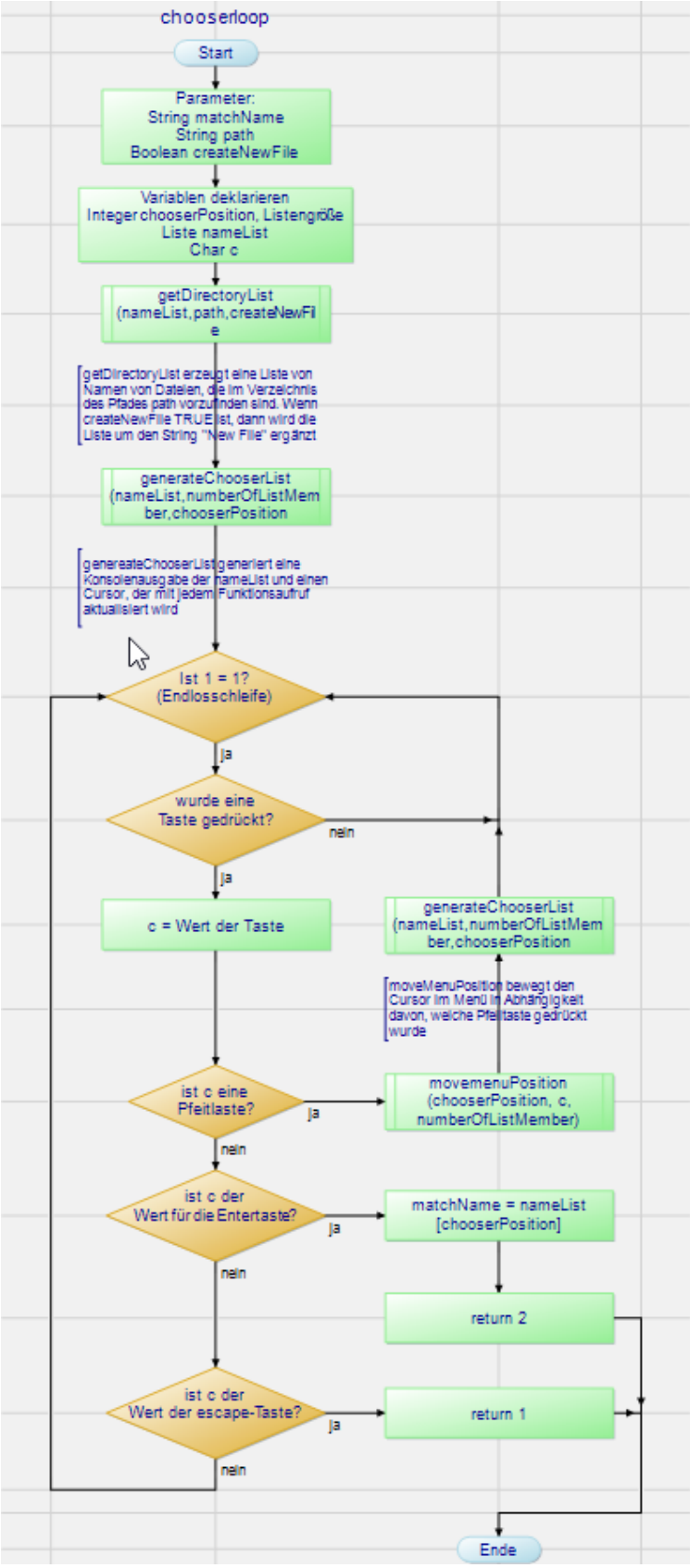

Abbildung 6

3.2 Prozessschritte, Vorgehensweise, Qualitätssicherung

Begonnen wurde zuerst mit der grafischen Darstellung des Sudokus. Es war die generateField-Funktion. Parallel dazu wurde die Navigation in dem Sudoku-Feld in der Funktion cursorLoop entwickelt. Nachdem der Grundbaustein geschaffen wurde, waren die nächsten Schritte das Entwickeln der Ladefunktion und aufbauend auf der cursorLoop-Funktion die Navigation in den Menüs.

Nach einem Abgleich und Austausch konnten wir die Funktionen im Team verwenden.

Als nächstes wurden noch einige Menügraphiken erstellt und damit alle benötigten Menüs fertiggestellt.

Nachdem die Lade-Funktionen implementiert wurde, konnte einerseits mit der Entwicklung des Überprüfungsalgorithmus und andrerseits die Arbeit an der Speicher-Funktion begonnen werden.

Im letzten großen Sprint wurde dann noch der Überprüfung- und der Lösungshinweisalgorithmus fertiggestellt.

3.3 Abweichungen, Anpassungen, Entscheidungen

Eine wichtige Entscheidung war es, dass wir für viele Sachen als Rückgabewert Defines festlegen, damit wir aussagekräftige Rückgabewerte besitzen. Besonders im gameLoop war die Übersicht um einiges klarer als dort die definierten Werte benutzt wurden.

4 Projektergebnisse

4.1 Soll-Ist-Vergleich

Die Anzeige des Gitters haben wir mit der generateField-Funktion gelöst. Diese zeigt nicht nur das Gitter an, sondern auch sämtliche Instruktionen, wie man das Programm bedienen soll.

Die Auswahl des aktuellen Feldes haben wir mit dem Cursorloop geregelt. Dieser steuert einen Cursor über das Spielfeld und wartet generell auf Eingaben des Users. Dadurch wird das gesamte Spiel gesteuert.

Die Auswahl des zu spielenden Sudokus läuft über die chooserloop-Funktion und der loadGameFromFile-Funktion. Der chooserloop navigiert den Benutzer über eine Oberfläche, mit der er eine Datei auswählen kann (alles vom Programm geführt). Die load-Funktion erstellt dann das entsprechende Spiel aus der Datei. Der Schwierigkeitsgrad ist am Dateinamen zu erkennen:
easy - für einfache Rätsel
medium - für mittelschwere Rätsel
hard - für schwere Rätsel

Die Erkennung einer korrekten Lösung haben wir mit der Funktion checkIfSolved umgesetzt.

Die Zeitmessung haben wir in diversen Funktionen einbauen müssen. Hier entstand die Heerausforderung das Speichern und Laden eines Spieles zu berücksichtigen.

Die Erstellung eines zufallsgenerierten Sudokus haben wir mit der Funktion randomGameGen realisiert. Diese Funktion hat diverse Unterfunktionen.

Das Speichern und Laden eines Spiels haben wir mit den Funktionen loadGameFromFile, saveGame und den Funktionen des FileChoosers realisiert. Das Einlesen eines Sudokus aus einer Datei ist an sich nichts anderes, als ein gespeichertes Blanko-Sudoku-Rätsel.

Den Lösungshinweis haben wir mit den Funktionen aus der randomGameGenerator.c realisiert.

4.2 Qualitätskontrolle

Die Steuerung der Menüs funktioniert auch mit P,K,H und M. Die Funktionen sind komplett kommentiert und "aufgeräumt". Es existiert noch eine "debug-Farbe" (Violett). Diese Farbe wurde dafür benutzt um Fehlerhafte Felder im Entwicklungsprozess darzustellen. Dementsprechend gibt es auch noch die Eigenschaft Error am Struct SudokuFeld. Außerdem können "nur" maximal 99 Dateien in die chooserlist eingelesen werden. Wenn ein Verzeichniss mehr Dateien aufweist, dann werden nur die ersten 99 aufgelistet. Die ist durch eine kleine Anpassung erweiterbar. Wir haben

es aber nicht als nötig empfunden. Zudem haben wir keine Funktion implementiert, mit der man Dateien löschen kann. Dies war allerdings auch nicht in den Anforderungen enthalten. Das Löschen geht trotzdem noch indem man manuell in das Verzeichniss "matchfields" oder "savegames" geht und die Dateien löscht.

4.3 Abweichungen, Anpassungen

Eingabemöglichkeiten sind bei uns die Zahlen von 0 bis 9, da wir die 0 als "leer" benutzen. Dementsprechend werden die Felder, die im Hintergrund 0 als Zahl gespeichert haben auch als 0 dargestellt.

4.4 Fazit

Alles in allem funktioniert das Programm stabil. Durch unsere Struktur ist das Programm auch jetzt noch leicht erweiterbar. Dadurch, dass wir uns relativ früh für ein struct entschieden haben, hatten wir auch kein Problem damit dieses Struct zu erweitern und somit Zusatz Anforderungen leichter zu bewältigen.

5 Anlagen

5.1 Quellverzeichnis

Farbdarstellung in der Konsole

http://www.cplusplus.com/forum/beginner/54360/[20.05.2018]

Bewegung des Cursors

https://www.computerbase.de/forum/showthread.php?t=202425[20.05.2018]
https://docs.microsoft.com/en-us/windows/console/using-the-high-level-input-and-output-functions[20.05.2018]

Auslesen eines Verzeichnisses

https://www.unixboard.de/threads/verzeichnis-auslesen-und-dateien-nummerieren-in-c.18416/[17. Juli 2007]

Dateien beschreiben und lesen

http://www.c-howto.de/tutorial/dateiverarbeitung/oeffnen-schliessen/[20.05.2018}

Windowssettings

https://forum.chip.de/discussion/1560510/console-size-veraendern
https://docs.microsoft.com/en-us/windows/console/setconsolewindowinfo

Backtrackingalgorithmus für den Solver

https://www.geeksforgeeks.org/backtracking-set-7-suduku/

5.2 Abbildungsverzeichnis

Im Paps-Ordner befinden sich die originalen Programmablaufpläne.

Nummer der Abbildung	Name	Kapitel
1	SaveLoadChooserloop-Datei	
2	SaveLoadChooserloop-Datei	
3	sodokuSolver-Datei	
4	checkifsolved-Datei	
5	gameloop-Datei	
6	SaveLoadChooserloop-Datei	