**4AHIT**

09.11.2013

**Sudoku**

**Protokoll**

**Ayvazyan Ari & Belinic Vennesa**

**Inhaltsangabe**

[Git-Link 1](#_Toc371187140)

[Aufgabenstellung 1](#_Toc371187141)

[Designüberlegung 2](#_Toc371187142)

[Arbeitsaufteilung 3](#_Toc371187145)

[Aufwandschätzung 3](#_Toc371187148)

[Endzeitaufteilung 3](#_Toc371187149)

[Arbeitsdurchführung 4](#_Toc371187150)

[Testbericht 5](#_Toc371187151)

[Quellenangaben 8](#_Toc371187152)

**Git-Link**

https://github.com/aayvazyan-tgm/Sudoku

# Aufgabenstellung

Erstellen Sie einen Algorithmus zur Lösung von Sudokurätseln. Dabei sollen nicht nur klassische Sudokus gelöst werden können, sondern zumindest ein ähnliches Rätsel (z.B. X-Sudoku oder Squiggly) [1]. Die Eingabe der Rätsel erfolgt zwingend über Dateien (csv) [2]. Optional kann es auch händisch über die Konsolenapplikation eingegeben werden (zusätzlich zum Ladevorgang aus einer Datei), wobei dann das Rätsel auch abspeicherbar sein soll.

Das Menü soll somit die gewählten zwei Rätselalgorithmen anzeigen, eine Lademöglichkeit anbieten und optional eine Eingabemöglichkeit.

Teamwork

Diese Aufgabenstellung MUSS von zwei Schülern gelöst werden, um das Trennen von Funktionen und Arbeitsteilung zu trainieren. Verwenden Sie auf jeden Fall selbsterstellte Headerdateien und lagern Sie zumindest den Berechnungsalgorithmus in eine eigene .c-Datei aus! Es muss ein Headerfile für das Kompilieren und Linken sowie Ausführen und Aufräumen bereitgestellt werden.

Abgabe

Es ist auf jeden Fall ein Protokoll über die Arbeitsschritte und die verwendete Logik zu erstellen. Dabei sind auch Testfälle zu beschreiben. Das Protokoll muss als PDF-Dokument den Sources beigegeben und als ZIP Archiv abgegeben werden.

# Designüberlegung

Sudoku

Da es erlaubt ist bereits bestehenden Sourcecode zu verwenden, hatte ich vor im Internet nach brauchbarem Sourcecode zu suchen (ein Pragramm in C, dass Sudokurätsel löst und mit Files arbeitet). Diese Prgramm würde ich dann so anpassen, dass es der Angabe entspricht.

X-Sudoku oder Squiggly

# Arbeitsaufteilung

Die Arbeit wird so aufgeteilt, dass jedes Gruppenmitglied, soweit wie möglich, von einander unabhängig arbeiten kann. Deshalb werden den jeweiligen Gruppenmitglieder ganze Programmteile als Arbeitspakete zugeteilt. Jedes Gruppenmitglied testes die von ihm/ihr geschriebenen Files so weit es nötig ist.

Vennesa

## Sudoku

Ari

## X-Sudoku oder Squiggly

|  |  |  |  |
| --- | --- | --- | --- |
|  | *geschätzte Zeit* | *tatsächliche Zeit* | *Kommentar* |
| **Ayvazyan Ari** |  |  |  |
| **Belinic Vennesa** | 3 h | 6 h | 1 h um den Code zu lesen und zu verstehen 3 h für das Ändern des Codes 2 h für die Dokuemntation |

# Aufwandschätzung

|  |  |  |
| --- | --- | --- |
| Arbeitspaket | | Aufwand in h |
| Sudoku | | 3 |
| X-Sudoku oder Squiggly | |  |
| SUMME |  | |

# Endzeitaufteilung

|  |  |  |
| --- | --- | --- |
| Arbeitspaket | | Aufwand in h |
| Sudoku | | 6 |
| X-Sudoku oder Squiggly | |  |
| SUMME |  | |

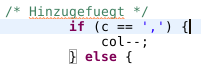
# Arbeitsdurchführung

Sudoku

Als erstes suchte ich im Internet nach bereits vorhandenen Algorithmen, um nicht "das Rad neu zu erfinden". Nach gründlicher Recherche stieß, auf einige brachbare Beispiel. Nachdem ich mir diese ein wenig durchgelesen hatte entschloss ich mich für ein Beispiel das mit txt-Files arbeitet, da ich annahm, dass ich bei diesem Beispiel nicht allzu große Änderungen vornehmen muss.

Ich nahm einige wenige Änderung im Code vor, damit dieser mit CSV-Files funktioniert:

Das Programm arbeitet standartmäßig mit txt-Files, für die noch nicht glösten Zahlen wurden Punkte eingesetzt. Ich änderte diesen Punkt auf 0, da bei mir der Benutzer 0 eingeben soll wenn die Zahl noch nicht gelöst ist.

Da in CSV-File einen Seperator git, um die Werte von einander zu trennen, muss dies berücksichtigt werden, in der Methode zum Einlesen. Dies befindet sich in der puzzle\_read-Funktion. Das c ist das aktuelle Zeichen, col ist die aktuelle Spalte.

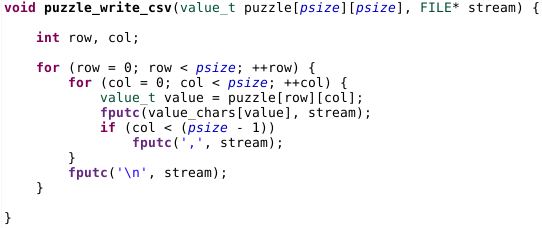
Hier habe ich zur puzzle\_solve Methode einenParameter stream vom Type FILE\* hinzugefügt, um die Lösung in ein CSV-File schreiben zu können.



Hier führe ich dann meine eigene Funktion puzzle\_write\_csv aus, hier verwende ich den Parameter den ich oben hinzugefügt habe.



Inhalt der puzzle\_write\_csv Funktion.



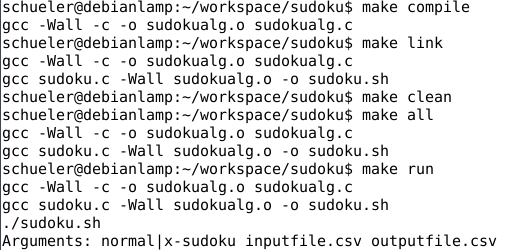
Letztlich trennte ich die Includes und Defines von den Funktionen und der Main Methode.

X-Sudoku oder Squiggly

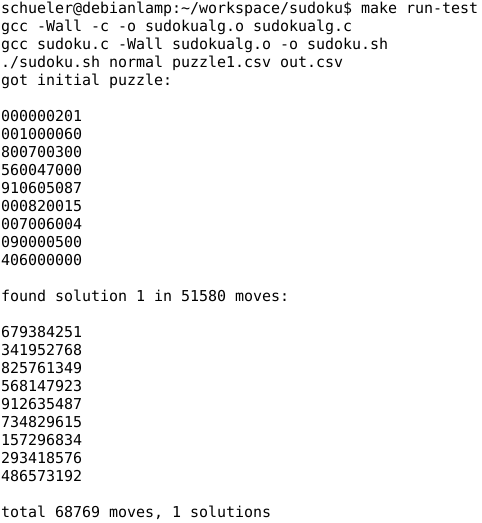
# Testbericht

Sudoku

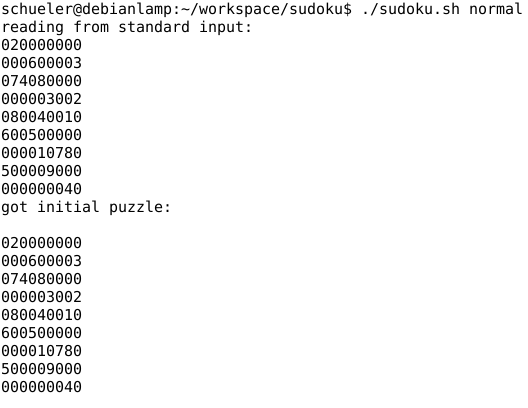
**Hier probiere ich die Targets des Makefiles aus:**

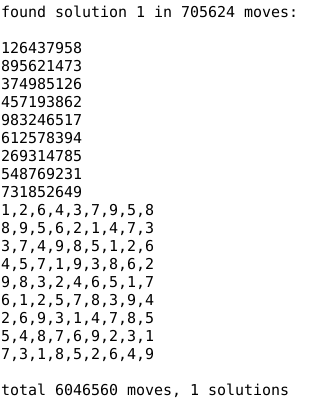


**Im Makefile ist noch einen Target enthalten, dass das Programm mit fixen Werten testet, diese führe ich hier ausführen:**

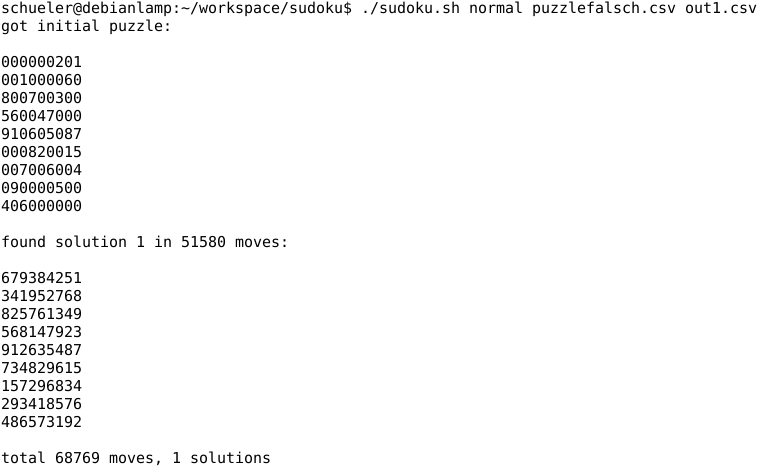


**Hier probiere ich das Programm mit dem Standartinput aus:**

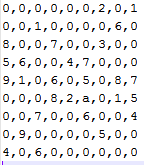




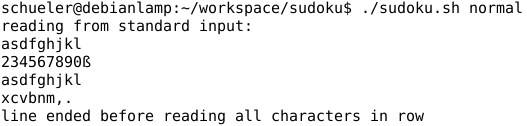
**Falls ein fehlerhafter Wert im Sudoku steht wird dieser it 0 eingelesen:**



**Inhalt von puzzlefalsch.csv**



**Über Standartinput einlesen, mit Fehlern:**



X-Sudoku oder Squiggly

# Quellenangaben

1. Matt Zucker, "Sudoku.c", zuletzt aktualisierte: 2010, online verfügbar: http://www.swarthmore.edu/NatSci/mzucker1/e15/sudoku.html, zuletzt besucht am: 30.10.2013