12.11.2013

Erstellen Sie einen Algorithmus zur Lösung von Sudoku Rätseln.

SYT - SUDOKU

Wolfgang Mair , Stefan Pitirut

Inhaltsangabe

[*Aufgabenstellung:* 2](#_Toc350533738)

[*Arbeitsschritte:* 2](#_Toc350533739)

[*Arbeitsaufteilung:* 2](#_Toc350533740)

[*Zeitaufwand:* 3](#_Toc350533741)

[*Designüberlegung:* 3](#_Toc350533742)

[*Arbeitsdurchführung:* 4](#_Toc350533743)

[*Technologienbeschreibung:* 4](#_Toc350533744)

[*Testdurchläufe:* 4](#_Toc350533745)

[*Quellenangaben:* 4](#_Toc350533746)

## *Aufgabenstellung:*

## SYT-INDINF-06 "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.  
  
**Links**[1] [Sudoku](http://en.wikipedia.org/wiki/Sudoku)[2] [Formatiertes Einlesen/Ausgeben von Streams mit fprintf() und fscanf()](http://openbook.galileocomputing.de/c_von_a_bis_z/016_c_ein_ausgabe_funktionen_009.htm#mje5a9730700783816c72f0f4556b59d10)

## *Arbeitsschritte:*

Sudoku

Solver

Exceptionhandling

Speichern/Laden

Makefile

Debugging

## *Arbeitsaufteilung:*

|  |  |  |
| --- | --- | --- |
|  | Mair | Pitirut |
| Sudoku | x |  |
| Solver |  | x |
| Exceptionhandling | x | x |
| Speichern/Laden | x |  |
| Makefile/Header-File | x | x |
| Debugging |  | x |

## *Zeitaufwand:*

|  |  |  |
| --- | --- | --- |
|  | Geschätzt | Tatsächlich |
| Sudoku |  |  |
| Solver | 30m | 30m |
| Exceptionhandling | 1h | 1h |
| Speichern/Laden |  |  |
| Makefile/Header-File |  |  |
| Debugging | 2h | 3h30 |
| **Gesamtzeit** |  |  |
| **Gesamtzeit** | **Mair** | **Pitirut 5h** |

## *Arbeitsdurchführung:*

Erfolge:

* Das Header-File war schneller erledigt als erwartet, auch das Makefile wurde in kürzerer Zeit erledigt als gedacht.
* Das trennen von Dateien ging einfacher als am Anfang erwartet.

Niederlagen:

* Das „returnen“ eines Arrays ist uns leider nicht gelungen!
* Debugging der Speichern-Methode war aufwendiger als der rest, das Programm wies Fehler auf, welche nicht vorhanden waren, somit zog sich das Debugging längern hin als erwartet!

## *Technologiebeschreibung:*

Makefiles:

Makefiles dienen dazu, um bei größeren Projekten, die Kompilierung von Quelltext-Dateien über das „make“ Programm zu steuern. Bei größeren Projekten besteht die Kompilierung aus vielen einzelnen Schritten und Abhängigkeiten. Diese werden in einem Makefile zusammengefasst, um sie im „make“ Programm zu verabeiten.

## *Testdurchläufe:*

## *Quellenangaben:*

[24] Daniel Brettschneider, zuletzt bearbeitet: 01.11.09 , zuletzt geöffnet: 13.11.13

<http://www.daniel-brettschneider.de/programmieren/sudoku-loesen-c>

[72] Elias Fischer, zuletzt bearbeitet: - , zuletzt geöffnet: 14.11.13

<http://www.c-howto.de/tutorial-makefiles.html>

[73] Elias Fischer, zuletzt bearbeitet:-, zuletzt geöffnet: 14.11.13

<http://www.c-howto.de/tutorial-makefiles-beispiel.html>