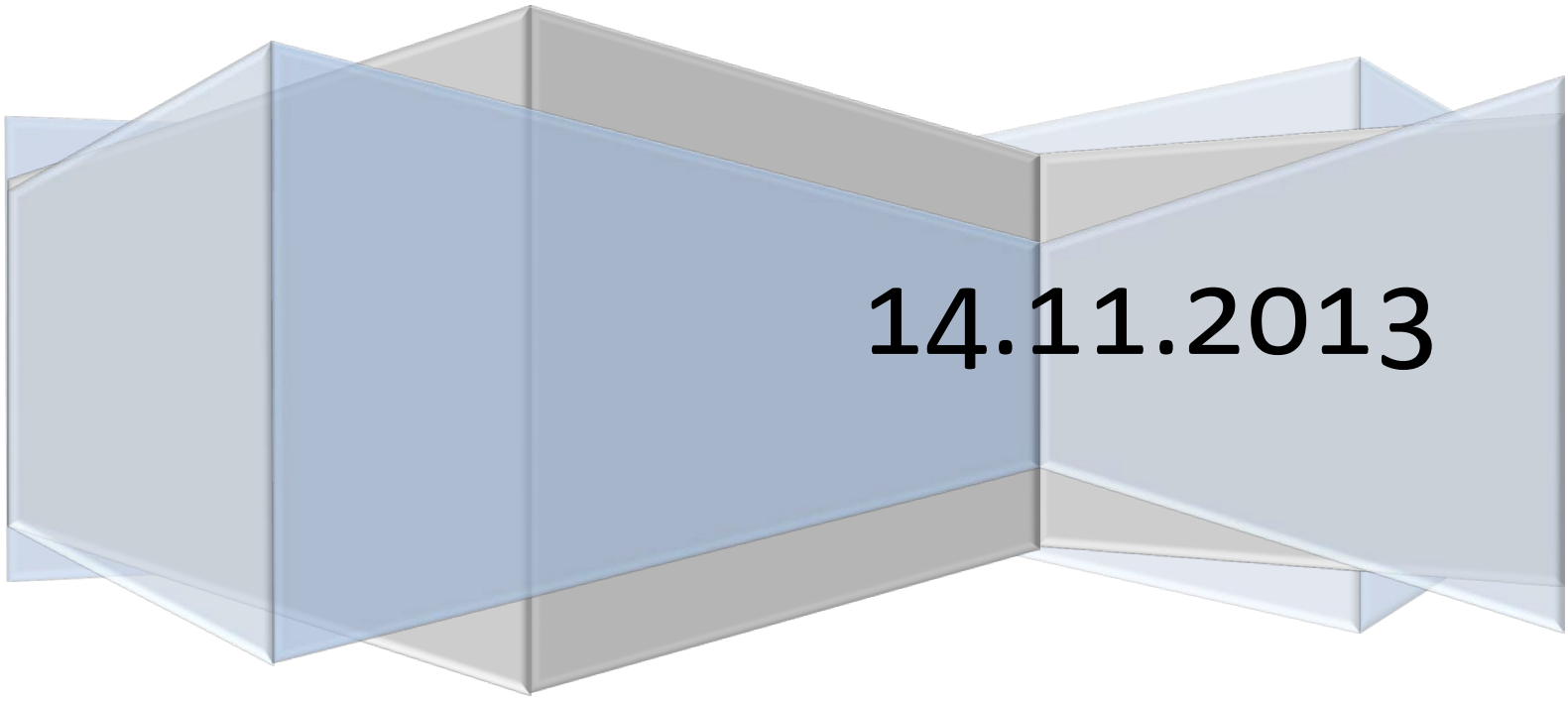
4YHIT

SUDOKU

INDINF A06: Sudoku

Elias Frantar, Gary Ye



14.11.2013

Inhaltsverzeichnis

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). Die Eingabe der Rätsel erfolgt zwingend über Dateien (csv). 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.

Aufwandschätzung:

Aus der folgenden Tabelle ist die Zeitplanung für dieses Projekt, auf wichtige Eckpunkte dieser Aufgabenstellung aufgeteilt, abzulesen.

Aufgabe	**Dauer in h**	**Anmerkungen**
Makefile	0,5	Einarbeitung und Erstellung
Sudokulösungsalgorihtmus	0,5	Suche und Anpassung
X-Sudokulösungsalgorithmus	0,5	Suche und Anpassung
Einlesen	1	Konsole und File
Ausgabe und Abspeicherung	1	Konsolenausgabe und Abspeicherung der Eingabe
Testing und Dokumentation	2	Protokoll verfassen, Testing
Gesamt	5,5	Abgabe in einem zip-Archiv

Arbeitsbeschreibung:

Die folgenden Schritte wurden in chronologischer Reihenfolge durchgeführt, wobei Schritt 10 während die komplette Durchführung begleitet hat. Alle „Aufgetretenen Probleme“ wurden durch zusätzlichen Zeitaufwand erfolgreich gelöst.

1. Erstellung der Projektstruktur und der notwendigen Headerfiles

2. Einarbeitung in Makefiles und Erstellung des für die vorher erstellte Projektstruktur funktionierende *makefile*

   Aufgetretene Probleme: automatisches Suchen und kompilieren von angegebenen Files durch Wildcards

3. Erstellung des main-files und Implementierung des Einlesens aus einem File und direkt von der Kommandozeileneingabe, sowie Abspeicherung in einem selbst definierten Sudoku-*struct*s

4. Implementierung der formatierten Ausgabe des Sudokus

5. Suche eines Lösungsalgorithmus für ein normales Sudoku und Anpassung dieses Algorithmus an unseren Sudoku-Datentyp

6. Abänderung des normalen Lösungsalgorithmus X-Sudoku-Lösungsalgorithmus, sodass damit auch ein X-Sudoku gelöst werden kann

   Aufgetretene Probleme: erfolglose Suche nach einem fertigen Lösungsalgorithmus

7. Implementierung der Abspeicherung eines händisch eingegebenen Sudokus

8. Testing der Algorithmen, sowie Eingabe und Ausgabe

9. Erstellung des Protokolls

10. Etwaige Anpassungen und Optimierungen der Dokumentation, der c-Files und des makefiles

In der folgenden Tabelle ist die für jeden Punkt aufgewendete Zeit, auf Personen aufgeteilt, abzulesen.

Aufgabe	**Frantar**	**Ye**	**Gesamt**
Makefile	35 min	15 min	50 min
Sudokulösungsalgorihtmus	25 min	10 min	35 min
X-Sudokulösungsalgorithmus	15 min	30 min	45 min
Einlesen	0 min	30 min	30 min
Ausgabe und Abspeicherung	30 min	0 min	30 min
Testing und Dokumentation	70 min	50 min	120 min
Verbesserung	10 min	35 min	45 min
Gesamt	~ 3 h	~ 3 h	~ 6 h

Neue Techonologien:

Headerfiles

In den Headerfiles stehen meistens die Prototypen von Funktionen, damit diese, wenn sie von einem anderen File inkludiert werden, aufgerufen werden können. Weiteres sollten sie auch structs, falls vorhanden, enthalten.

Wir haben ein Sudoku struct in einem Headerfile namens „sudoku.h“ definiert. Dieses wurde in jedem File, das diesen Datentyp benötigt, inkludiert. Dies erspart erneutes Definieren in jedem einzelnen File.

```
#ifndef SUDOKU_H_INCLUDED
#define SUDOKU_H_INCLUDED

typedef struct{
  int grid[9][9];
  int type;
} sudoku;

#endif
```

Anzumerken ist, dass jedes Header File einen #include Guard besitzt, um mehrfaches Inkludieren zu vermeiden.

Eingabe/Ausgabe via fprintf(), fscanf()

Der einzige Unterschied zwischen scanf()/printf() und fscanf()/printf() liegt darin, dass die zweiteren Funktionen in Files schreiben bzw. aus Files lesen können. Möchte man eine scanf()/printf() äquivalente Funktion mit fscanf()/fprintf() aufrufen, so gibt man als ersten Parameter (FILE*) stdin/stdout an.

Beispiel:

```
int a;
// Folgende zwei Codezeilen haben denselben Effekt
scanf("%d", &a);
fscanf(stdin, "%d", &a);
```

Doch möchte man mit einem richtigen File interagieren, so erstellt man einen FILE*, der später für *fprintf()/fscanf()* verwenden kann, mittels der Funktion fopen(), die man auch in der <stdio.h>-Library vorfinden kann. Sie benötigt als Parameter, den Filenamen und die Operation, die an dem File durchgeführt werden soll (Lesen „r“ / Schreiben „w“).

Achtung: Man sollte darauf achten, dass die alten Daten verloren gehen, falls bereits ein File mit demselben Filename existiert.

Testing:

Lösungsalgorithmen:

Zum Testen der beiden Lösungsalgorithmen werden jeweils ein X-Sudoku und ein normales Sudoku in .in-Files geschrieben. Anschließend wird das Programm mit diesen Files gestartet.

Sudoku:

```
Enter filename: sudoku.in

+-------+-------+-------+
| 0 1 0 | 0 4 0 | 6 0 0 |
| 0 0 0 | 0 0 8 | 0 0 0 |
| 8 0 4 | 9 0 2 | 0 0 7 |
+-------+-------+-------+
| 1 5 0 | 0 0 0 | 0 0 2 |
| 0 0 8 | 3 0 5 | 9 0 0 |
| 4 0 0 | 0 0 0 | 0 7 5 |
+-------+-------+-------+
| 9 0 0 | 4 0 6 | 2 0 3 |
| 0 0 0 | 8 0 0 | 0 0 0 |
| 0 0 2 | 0 9 0 | 0 5 0 |
+-------+-------+-------+
+-------+-------+-------+
| 2 1 3 | 5 4 7 | 6 9 8 |
| 5 9 7 | 1 6 8 | 3 2 4 |
| 8 6 4 | 9 3 2 | 5 1 7 |
+-------+-------+-------+
| 1 5 9 | 6 7 4 | 8 3 2 |
| 7 2 8 | 3 1 5 | 9 4 6 |
| 4 3 6 | 2 8 9 | 1 7 5 |
+-------+-------+-------+
| 9 7 1 | 4 5 6 | 2 8 3 |
| 3 4 5 | 8 2 1 | 7 6 9 |
| 6 8 2 | 7 9 3 | 4 5 1 |
+-------+-------+-------+
```

X-Sudoku:

```
Enter filename: xsudoku.in

+-------+-------+-------+
| 0 1 0 | 8 0 0 | 6 0 5 |
| 7 8 0 | 4 0 0 | 0 0 0 |
| 0 2 0 | 0 6 1 | 0 0 0 |
+-------+-------+-------+
| 0 3 0 | 1 0 0 | 0 0 8 |
| 0 0 5 | 0 7 0 | 0 0 0 |
| 4 9 0 | 0 0 0 | 0 0 0 |
+-------+-------+-------+
| 0 5 0 | 0 0 9 | 0 0 0 |
| 8 0 0 | 5 0 3 | 0 0 0 |
| 0 0 0 | 0 0 0 | 0 0 2 |
+-------+-------+-------+
+-------+-------+-------+
| 3 1 4 | 8 9 2 | 6 7 5 |
| 7 8 6 | 4 3 5 | 1 2 9 |
| 5 2 9 | 7 6 1 | 3 8 4 |
+-------+-------+-------+
| 2 3 7 | 1 5 4 | 9 6 8 |
| 1 6 5 | 9 7 8 | 2 4 3 |
| 4 9 8 | 3 2 6 | 7 5 1 |
+-------+-------+-------+
| 6 5 1 | 2 4 9 | 8 3 7 |
| 8 7 2 | 5 1 3 | 4 9 6 |
| 9 4 3 | 6 8 7 | 5 1 2 |
+-------+-------+-------+
```

Wie man leicht erkennen kann, wurden beide Sudokus in kaum wahrnehmbarer Zeit korrekt gelöst.

Eingabe:

Nun wird getestet, ob die Eingabe sowohl per File als auch per manueller Eingabe funktioniert.

- **Einlesen aus einem File:**

```
0: Sudoku 9x9
1: Sudoku X
0
0: input by file
1: manual input
0
Enter filename: sudoku.in
```

Das in „sudoku.in“ gespeicherte Rätsel wird nach dieser Eingabe korrekt gelöst.

- **Manuelle Eingabe:**

```
0: Sudoku 9x9
1: Sudoku X
0
0: input by file
1: manual input
1
0,1,0,0,4,0,6,0,0
0,0,0,0,0,8,0,0,0
8,0,4,9,0,2,0,0,7
1,5,0,0,0,0,0,0,2
0,0,8,3,0,5,9,0,0
4,0,0,0,0,0,0,7,5
9,0,0,4,0,6,2,0,3
0,0,0,8,0,0,0,0,0
0,0,2,0,9,0,0,5,0
```

Das manuell eingegebene Rätsel wird nach dieser Eingabe korrekt gelöst.

Abspeichern:

Jetzt muss noch getestet werden, ob das Abspeichern eines manuell eingegebenen Rätsels funktioniert.

```
0: Sudoku 9x9
1: Sudoku X
0
0: input by file
1: manual input
1
0,1,0,0,4,0,6,0,0
0,0,0,0,0,8,0,0,0
8,0,4,9,0,2,0,0,7
1,5,0,0,0,0,0,0,2
0,0,8,3,0,5,9,0,0
4,0,0,0,0,0,0,7,5
9,0,0,4,0,6,2,0,3
0,0,0,8,0,0,0,0,0
0,0,2,0,9,0,0,5,0
Do you want to save this Sudoku? (y|n)
y
Enter filename: sudoku_test.in
```

Nach dieser Eingabe ist im aktuellen Verzeichnis ein File „sudoku_test.in“ zu finden, welches das korrekt abgespeicherte Rätsel enthält.