UNIVERSITÄT OSNABRÜCK

Dr. Thomas Wiemann

2. Übungsblatt zur Vorlesung „Einführung in die Programmiersprache C++“

Konzepte dieses Aufgabenblattes:

In diesem Aufgabenblatt werden sie sich mit C-Arrays und Pointern auseinandersetzen. Im ersten Teil des Aufgabenblattes werden sie einen rekursiven Algorithmus zum Lösen eines Sodoku-Rätsels implementieren, das als zweidimensionales C-Array repräsentiert wird. Zudem werden sie lernen, wie in C-Programmen die Schnittstellendefinition von der Implementierung getrennt wird, indem sie ihr erstes Header-File schreiben.

Aufgabe 2.1: Sodoku-Solver (70 Punkte)

In dieser Aufgabe werden Sie ein Programm zur Lösung eines Sudoku-Puzzles schreiben. Dabei werden Sie etwas über Rekursion, Backtracking und den Umgang mit zweidimensionalen Arrays lernen.

Problemstellung:
Gegeben sei folgendes Sudoku:

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

Beschreibung des Lösungsalgorithmus:

Schreiben Sie ein C-Programm, welches das gegebene Puzzle löst. Repräsentieren Sie das Spielfeld als zweidimensionales Integer-Array, in dem die freien Stellen mit 0 markiert sind. Lösen Sie das Problem mit rekursiven Backtracking: Testen sie nacheinander für jedes freie Feld systematisch beginnend mit 1, ob es eine konfliktfreie Belegung gibt. Gibt es eine, versuchen Sie das nächste freie Feld, wieder mit 1 beginnend, zu belegen usw. Sollte es zu einem Punkt keine konfliktfreie Belegung mehr geben, nehmen Sie Ihre letzte Entscheidung zurück und versuchen, beginnend mit der Ziffer nach der alten Belegung, wieder eine Lösung zu finden. Natürlich kann es vorkommen, dass auch mehrere Züge zurückgenommen werden müssen. Welchem aus Ihrer Informatik-Einführung bekannten Suchverfahren entspricht dieses Vorgehen? Kann man damit für jedes lösbare Sudoku eine Lösung finden? Wie viele Belegungen werden im schlimmsten Fall getestet?

Programmbeschreibung:

Implementieren Sie die folgenden Funktionen:

```
int is_valid(int z, int i, int j, int sudoku[9][9])
```

Diese Funktion soll "wahr" zurückgeben, wenn die Ziffer z an Position (i, j) im Spielfeld konfliktfrei gesetzt werden kann.

```
int solve_sudoku(int i, int j, int sudoku[9][9])
```

Diese Funktion soll zur rekursiven Lösung des Rätsels verwendet werden. Sie gibt "wahr" zurück, falls ausgehend vom Feld (i, j) eine Lösung gefunden werden kann.

```
void print_sudoku(int sudoku[9][9])
```

Diese Funktion gibt das durch das Array `sudoku` repräsentierte Spielfeld formatiert auf der Konsole aus. Das Spielfeld soll hier hardgecoded sein, d.h., sie brauchen keine Funktion zum Einlesen des Puzzles schreiben.

Lagern Sie die diese Definitionen in einen eigenen Header `SodukuFunctions.h` aus. Implementieren Sie die entsprechenden Funktionen in einer Datei `SodokuFunctions.c`. Übersetzen sie diese Datei zu einer Bibliothek, die sie gegen ihr Hauptprogramm, das sich in der Datei `SodokuSolver.c` befindet zu einer Executable `sodoku_solver`. Schreiben sie dazu ein entsprechendes `Makefile`.

Geben sie nach dem Aufruf von `sodoku_solver` das gegebene Feld und die mittels Backtracking berechnete Lösung auf der Konsole aus. Kommentieren Sie Ihren Quellcode ausreichend mit aussagekräftigen Kommentaren. Dokumentieren Sie die Signaturen der von Ihnen implementierten Funktionen vollständig im Doxygen-Style. Die daraus erzeugbare Doku muss nicht mit abgegeben werden. Stellen Sie sicher, dass Ihr Programm den Style-Richtlinien der Vorlesung entspricht (siehe entsprechendes Dokument in StudIP). Nutzen Sie ausschließlich C-Funktionen. Übersetzen Sie Ihr Programm mit den Compiler-Parameter `-Wall`, um alle relevanten Warnungen angezeigt zu bekommen. Stellen Sie sicher, dass ihre Abgabe durch den Aufruf von `make` vollständig erzeugt wird.

Aufgabe 3.2: Pointerarithmetik *(30 Punkte)*

Neue Konzepte dieser Aufgabe: Operationen mit Pointer-Variablen und Arrays

Es seien `feld` ein `int`-Array `p`, `p1`, `p2` `int`-Zeiger und `i` eine `int`-Variable. Welche der folgenden Zuweisungen sind zulässig und welche nicht? Begründen Sie ihre Antwort!

a) `p = feld;`
b) `feld = p;`
c) `p = &feld[3];`
d) `feld[2] = p[5];`
e) `p1 = p2 + i;`
f) `p1 = i + p2;`
g) `i = p1 * p2;`
h) `i = p1 - p2;`
i) `i = p1 + p2;`

Erklären sie darüber hinaus, wie der `[][]`-Zugriff bei zweidimensionalen Arrays ohne die Verwendung von `[]` – d.h. nur über die Verwendung der Pointer– realisiert werden kann. Eine nette Möglichkeit verwirrenden Programmcode zu generieren ist es, Array-Namen und Index zu vertauschen, d.h. `feld[i]` stellt denselben Zugriff wie `i[feld]` dar. Warum ist das so? Geben sie Ihre Antworten und Begründungen schriftlich ab

Abgabe

Checken sie ihre Abgabe bis Montag, 05.11.2018, 08:00 Uhr in den Ordner `uebung02` des Master-Branch Ihres git-Repositories ein. Bringen Sie pro Gruppe einen Ausdruck der von ihnen erzeugten Dateien mit zum Testat.