Freiwillige Zusatzaufgabe: Sudoku

1 Einleitung

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

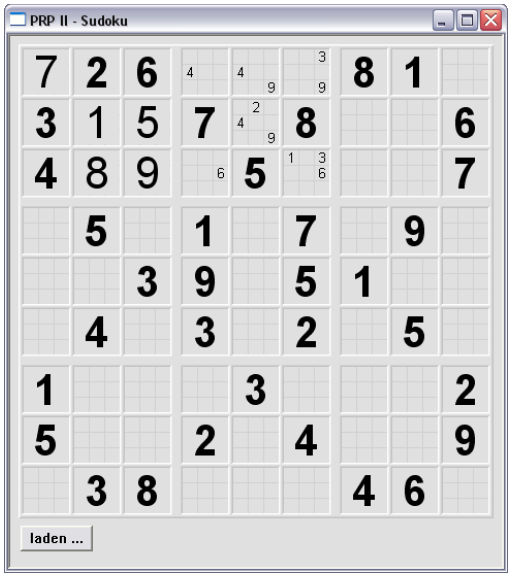
1.1 Das Spiel Sudoku

Sudoku ist ein altes Zahlenrätsel ähnlich dem magischen Quadrat, das sich seit einigen Jahren auch in Deutschland großer Beliebtheit erfreut. Das Wort *Sudoku* stammt aus dem Japanischen und bedeutet wörtlich „Eine Zahl bleibt immer allein“. Am weitesten verbreitet ist das 9er-Sudoku mit 9 mal 9 Feldern. Die Regeln sind einfach:

- In jedem Feld darf nur eine Ziffer von eins bis neun stehen
- In jeder Zeile muss jede Ziffer genau einmal erscheinen.
- In jeder Spalte muss jede Ziffer genau einmal erscheinen.
- In jedem 9er Block (3 mal 3 Felder) muss jede Ziffer genau einmal erscheinen.

Im Sudoku werden einige Zahlen vorgegeben; die restlichen Ziffern müssen gefunden werden.

1.2 Bedienung des Spiels



In diesem Praktikum soll ein Programm erstellt werden, mit dem der Benutzer komfortabel Sudoku spielen kann.

Damit sich der Benutzer mögliche Lösungen merken kann wird jedes der 81 Felder in 9 Elemente unterteilt. Klickt er auf eines der Elemente, so wird dort die zugehörige Ziffer ein- oder ausgeblendet.

Wird auf eines der Elemente in einem Feld ein Doppelklick ausgeführt, so wird die zugehörige Zahl als Lösung eingetragen. Die vom Benutzer gefundenen Zahlen werden etwas dünner als die fest vorgegebenen Zahlen eingetragen.

1.3 Programmgerüst

Für diese Aufgabe wurde ein Programmgerüst erstellt, welches die Interaktion mit dem Benutzer übernimmt. Es handelt sich dabei um eine Windows-Anwendung (mit WIN32 erstellt), die, wie unter Windows gewohnt, per Maus bedient wird. Das Gerüst kann auf der Internetseite www.rrhess.de heruntergeladen werden.

Gegenüber einer Konsole-Anwendung hat eine Windows-Anwendung einen wesentlichen Unterschied: Es ist nicht mehr die Anwendung, die den Computer steuert, sondern die Anwendung wird vom Betriebssystem gesteuert. Das hat zur Folge, dass die Anwendung auf alle möglichen Anfragen reagieren muss. Anfragen können z.B. sein: Fenster zeichnen, auf einen Mausklick oder Tastendruck reagieren, Anwendung schließen etc.

Das hier vorgestellte Gerüst besteht aus drei Dateien:

- `main.c` beinhaltet alle Programmstücke für die Interaktion mit dem Benutzer.

- `sudoku.h` ist die Header-Datei mit einigen Deklarationen für das Programm.
- `sudoku.c` schließlich beinhaltet die Logik des Sudoku-Spiels.

Die Dateien `main.c` und (mit nur einer Ausnahme) `sudoku.h` brauchen von Ihnen nicht verändert zu werden. Ihre Arbeit beschränkt sich auf die Datei `sudoku.c`.

1.4 Die Funktion `Sudoku()`

Herzstück des Programms ist die Funktion Sudoku(), die von Ihnen mit Leben gefüllt werden soll. Sie wurde wie folgt deklariert:

```
int Sudoku(tSudokuTask task, int a, int b, int c);
```

Der erste Parameter ist eine `enum`-Variable Namen `task` für die zu erledigende Aufgabe. Je nach Wert des Parameters `task` muss die Funktion `Sudoku()` auf eine Frage reagieren:

task	**Beschreibung**
`getNumber`	Frage nach der gefundenen oder vorgegebenen Zahl des Feldes. Wurde die Zahl noch nicht gefunden, so kann der Rückgabewert beliebig sein.
`getGuess`	Frage, ob in einem Feld eine bestimmte Zahl geraten wurde
`getFinished`	Frage, ob die Zahl in einem Feld schon gefunden wurde
`getProtected`	Frage, ob die Zahl in einem Feld ein Vorgabewert ist
`singleClick`	Reaktion auf einen einfachen Mausklick auf ein Element in einem Feld
`doubleClick`	Reaktion auf einen doppelten Mausklick auf ein Element in einem Feld
`loadFile`	Ein Sudoku aus einer Datei landen
`saveFile`	Ein Sudoku in eine Datei speichern (optional)
`special1`	Kann von Ihnen definiert werden (optional)
`special2`	Kann von Ihnen definiert werden (optional)
`special3`	Kann von Ihnen definiert werden (optional)

Die Bedeutung der anderen drei Parameter und des Rückgabewerts ergibt sich aus der Aufgabe:

task	**a**	**b**	**c**	**Rüchgabewert**
`getNumber`	Spalte	Zeile	0	gefundene Ziffer (1-9)
`getGuess`	Spalte	Zeile	Element	ob Zahl geraten wurde (0/1)
`getFinished`	Spalte	Zeile	0	ob die Zahl gefunden wurde (0/1)
`getProtected`	Spalte	Zeile	0	ob die Zahl ein Vorgabewert ist (0/1)
`singleClick`	Spalte	Zeile	Element	nichts
`doubleClick`	Spalte	Zeile	Element	nichts
`loadFile`	Dateiname	0	Handle	nichts
`saveFile`	Dateiname	0	Handle	nichts
`special1`	0	0	Handle	nichts
`special2`	0	0	Handle	nichts
`special3`	0	0	Handle	nichts

Bedeutung	**Beschreibung**
Spalte	Spalte im Sudoku (0-8)
Zeile	Zeile im Sudoku (0-8)
Element	Element innerhalb eines Feldes im Sudoku (0-8)
Dateiname	Zeiger auf Zeichenkette (mit (char*) umwandeln)
Handle	Handle des Fensters (mit (HWND) umwandeln)

Da die Funktion zwischen den Aufrufen immer wieder verlassen wird, geht der Wert der lokalen Variablen verloren. Um den Wert einiger Variablen zu erhalten, müssen diese mit `static` definiert werden. Für diese statischen Variablen wird einmal Speicher reserviert und während der gesamten Laufzeit des Programms nicht wieder freigegeben.

In diesem Praktikum soll nur diese Funktion `Sudoku()` bearbeitet werden.

2 Pflichtaufgaben

2.1 Nötige Strukturen und Vektoren erstellen

Für jedes der 81 Felder im Sudoku müssen folgende Informationen gespeichert werden:

- gefundener Zahlenwert (1-9)
- Markierung ob Zahl gefunden wurde
- Markierung, ob die Zahl ein Vorgabewert ist
- Markierung, ob die Zahl 1 geraten wurde
- Markierung, ob die Zahl 2 geraten wurde
- ...
- Markierung, ob die Zahl 9 geraten wurde

Deklarieren Sie für ein Feld im Sudoku mit `struct` die passende Struktur. Verwenden Sie bevorzugt eine bitweise Struktur. In jedem Fall sollten Sie nicht unnötig Speicher verschwenden.

Definieren Sie mit der erstellten Struktur einen zweidimensionalen Vektor mit 9 mal 9 Elementen. Definieren Sie den Vektor mit `static`, damit die Daten zwischen den Aufrufen der Funktion erhalten bleiben.

Initialisieren Sie den erstellten Vektor beim ersten Aufruf der Funktion `Sudoku()` mit dem auf Seite 1 oben gezeigten Sudoku.

2.2 Interaktion mit dem Benutzer

Jetzt soll das Programm mit Leben erfüllt werden. Erstellen Sie dafür den Programmcode für die ersten sechs Tasks. Für die ersten vier, `getNumber`, `getGuess`, `getFinished` und `getProtected`, muss die entsprechende Information aus dem erstellten Vektor zurückgegeben werden. Für `singleClick` und `doubleClick` muss der Vektor entsprechend modifiziert werden.

2.3 Sudoku aus Datei lesen

Dem Projektgerüst sind drei Sudokus beigefügt. Es handelt sich dabei um Textdateien mit der Endung `sdk`. Die Felder sind zeilenweise eingetragen, wobei der Zahlenwert null einer leeren Zelle und die Ziffern eins bis neun den jeweiligen Vorgabewerten entsprechen. Das Beispiel rechts zeigt den Inhalt der Datei `Sudoku-001.sdk` mit dem Sudoku von Seite 1.

```
026000810
300708006
400050007
050107090
003905100
040302050
100030002
500204009
038000460
```

Ergänzen Sie in der Funktion `Sudoku()` den entsprechenden Programmcode um die beigefügten Sudokus lesen zu können.

Eventuelle Fehler beim Öffnen oder Lesen der Datei können Sie dem Benutzer über eine Nachrichtenbox mit der Funktion `MessageBox()` mitteilen. Diese Funktion erwartet als ersten Parameter den Handle auf das Fenster der Applikation, als zweiten Parameter den eigentlichen Text

der Nachricht, als dritten Parameter den Titel des Fensters und als vierten Parameter die Art der Nachrichtenbox. Verwenden Sie das Muster in dem Programmgerüst und passen Sie den zweiten und dritten Parameter an. Um einer Fehlermeldung mehr Nachdruck zu verleihen, können Sie als vierten Parameter statt `MB_OK` auch `MB_ICONEXCLAMATION` eintragen.

3 Optionale Aufgaben

Für die Aufgaben in diesem Abschnitt benötigen Sie weitere Tasten in Ihrem Programm. In dem Programmgerüst sind vier weitere Tasten vorgesehen, die sie über die Header-Datei `Sudoku.h` aktivieren können. Entfernen Sie dafür vor den entsprechenden `#define`-Zeilen die Kommentarzeichen. Für die drei Spezialtasten können Sie auch einen beliebigen Text eintragen, der dann auf den Tasten erscheint.

3.1 Sudoku in Datei speichern

Aktivieren Sie die Taste zum Speichern einer Datei. Erstellen Sie den Programmcode zum Speichern eines Sudokus. Speichern Sie dabei nur die gefundenen Zahlen mit eins bis neun, und die nicht gefunden Zahlen mit null. Die vermuteten Werte werden ignoriert.

3.2 Prüfen des Sudokus

Aktivieren Sie einen der optionalen Tasten und geben Sie ihr den Text „Test“ oder ähnlich. Erstellen Sie eine Funktion, die das bisherige Sudoku prüft. Es gibt dabei drei mögliche Ergebnisse: Sudoku fehlerhaft, Sudoku korrekt gelöst und Sudoku noch nicht fertig, aber bisher korrekt.

3.3 Sudoku lösen

Aktivieren Sie einen der optionalen Tasten und geben Sie ihr den Text „Lösen“ oder ähnlich. Erstellen Sie eine Funktion, die das Sudoku löst. Es gibt dabei drei mögliche Ergebnisse: Sudoku eindeutig gelöst, Sudoku nicht lösbar und Sudoku gelöst, aber es gibt mehrere Lösungen. Hinweis: Mit Rekursion lässt sich das Sudoku lösen.

Viel Erfolg beim Programmieren!