

Daten umformen

Mogeln mit EVA



Eingabe, Verarbeitung, Ausgabe: Das EVA-Prinzip durchdringt die gesamte Softwareentwicklung. Eine Analyse der Datenstrukturen und die Verwendung der passenden Algorithmen spielen dabei eine herausragende Rolle. Stefan, fällt dir dazu eine Übung ein?

Wer übt, gewinnt

dnpCode: A1006dojo

In jeder dotnetpro finden Sie eine Übungsaufgabe von Stefan Lieser, die in maximal drei Stunden zu lösen sein sollte. Wer die Zeit investiert, gewinnt in jedem Fall – wenn auch keine materiellen Dinge, so doch Erfahrung und Wissen.

Es gilt:

- Falsche Lösungen gibt es nicht. Es gibt möglicherweise elegantere, kürzere oder schnellere Lösungen, aber keine falschen.
- Wichtig ist, dass Sie reflektieren, was Sie gemacht haben. Das können Sie, indem Sie Ihre Lösung mit der vergleichen, die Sie eine Ausgabe später in dotnetpro finden.

Übung macht den Meister. Also – los geht's. Aber Sie wollten doch nicht etwa sofort Visual Studio starten ...



Wer kennt nicht *Minesweeper*, das beliebte Spiel, welches zum Lieferumfang von Windows gehört? Doch keine Sorge, es geht dieses Mal nicht wieder um eine aufwendige Benutzerschnittstelle, sondern um eine kleine Kommandozeilenanwendung. Die soll aus einer Eingabedatei eine Ausgabedatei erzeugen. Die Eingabedatei enthält die Beschreibung eines *Minesweeper*-Spielfeldes. Das Programm erzeugt als Ausgabedatei einen dazu passenden „Mogelzettel“. Der Aufruf erfolgt folgendermaßen:

```
mogelzettel spiel1.txt mogelzettel1.txt
```

Der Aufbau der Eingabedatei ist wie folgt: Die erste Zeile enthält die Anzahl der Zeilen und Spalten des Spielfeldes. Beide Zahlen sind durch ein Leerzeichen getrennt. Die nachfolgenden Zeilen enthalten dann jeweils die Konfiguration einer Zeile des Spielfeldes. Dabei sind freie Felder durch einen Punkt dargestellt, Felder mit einer Mine durch einen Stern. Hier ein Beispiel:

```
5 6
...*.
...*.
.....
*....*
.*....
```

Für diese Eingabedatei soll eine Ausgabedatei erzeugt werden. Die Ausgabedatei gibt für jedes Feld die Anzahl der Minen in der unmittelbaren Nachbarschaft an. Jedes Feld hat maximal acht Nachbarn, folglich können maximal acht Minen in der Nachbarschaft eines Feldes vorkommen, am Rand des Spielfeldes sind es natürlich weniger. Felder, die selbst eine Mine enthalten, sollen mit der Ziffer 0 belegt sein, es sei denn, in der Nachbarschaft befinden sich Minen. Dann sollen diese ebenfalls gezählt werden.

Der Mogelzettel gibt also keine direkte Auskunft darüber, wo die Minen liegen, sondern nur über die Anzahl der Minen in den jeweils benachbarten Feldern.

Für das obige Beispiel soll folgende Ausgabedatei erzeugt werden:

```
001211
001121
111121
121010
211011
```

Sie dürfen davon ausgehen, dass die Eingabedatei im korrekten Format vorliegt. Geht in der Folge etwas schief, ist gegebenenfalls das Ergebnis inkorrekt, oder das Programm bricht sogar ab. Dies soll hier keine Rolle spielen.

Die Vorgehensweise bei der Lösung dieser Übung dürfte Lesern der vorhergehenden Übungen inzwischen geläufig sein. Zunächst sollten die Anforderungen geklärt werden. Dazu ist es sinnvoll, Beispiele aufzuschreiben. Eines habe ich Ihnen gegeben, vielleicht definieren Sie weitere, um beispielsweise Randfälle zu definieren. Da ich Ihnen als „Kunde“ nicht so ohne Weiteres für Rückfragen zur Verfügung stehe, treffen Sie gegebenenfalls selber sinnvolle Annahmen. Nach der Klärung der Anforderungen beginnt die Planung. Dieser Phase sollten Sie viel Aufmerksamkeit und Zeit schenken. Je intensiver Sie sich mit dem Problem und seiner Lösung auseinandersetzen, desto besser sind Sie vorbereitet für die Implementierung. Zerlegen Sie das Gesamtproblem in kleinere Teilprobleme, suchen Sie mögliche Flows.

Wenn Sie sich bei der Herangehensweise nicht sicher sind, ob eine Idee tatsächlich zur Lösung des Problems führen wird, lohnt es sich, einen *Spike* zu erstellen. Ein *Spike* dient dazu, eine Idee zu überprüfen oder eine Technik oder Technologie zu explorieren. Code, der bei einem *Spike* entsteht, wird nicht produktiv verwendet, daher sind keine Tests erforderlich, und auch gegen Prinzipien darf gern verstoßen werden. Es geht um den Erkenntnisgewinn. Es könnte sich dabei schließlich auch herausstellen, dass eine Idee nicht zum Ziel führt. Im Anschluss an den *Spike* beginnen Sie dann testgetrieben mit der Entwicklung des Produktionscodes. Dabei beachten Sie selbstverständlich alle Prinzipien und Praktiken.

Und nun frisch ans Werk! Mogeln Sie aber bitte nur beim *Minesweeper*-Spielen, nicht bei der Softwareentwicklung. [ml]