

Übung 9

Game of Life

Thomas Klütz & Nadine Schärmann

Conway's "Game of Life" von 1970 entspricht einem abstrakten Modell für die Ausbreitung eines Organismus (z. B. Bakteriumwachstum). In einem zweidimensionalen Gitter kann jede Zelle entweder den **Zustand 1 (besetzt bzw. lebend)** oder **0 (leer bzw. ausgestorben)** annehmen. Die Nachbarschaft einer Zelle besteht aus den 8 direkt angrenzenden Zellen. Für den Übergang von einem Zeitschritt zum nächsten liegen die folgenden Regeln vor:

- **Überleben:** jede besetzte Zelle mit zwei oder drei lebenden Nachbarn lebt weiter.
- **Aussterben:** infolge von Überpopulation stirbt jede Zelle mit vier oder mehr Nachbarn und eine Zelle mit einem oder keinem Nachbarn stirbt aufgrund von Isolation.
- **Wachstum:** Jede leere Zelle mit genau drei lebenden Nachbarn wird zu einer lebenden Zelle.

Aufgabe: Wir werden hier den Organismus mit einem INTEGER-Feld simulieren. Um die Arbeit mit Teilfeldern einfacher zu gestalten und Fallunterscheidungen zu vermeiden, gibt es um das "aktive Feld" herum einen mit Nullen besetzten Rand (siehe Aufgabe 4).

1. Schreibe eine Funktion, welche einen einzelnen Zeitschritt simuliert. Der Funktion wird ein INTEGER-Feld (das Gitter mit einem Rand aus 0-Einträgen) als Argument übergeben. Für den Übergang zum nächsten Zeitschritt kann ein Teilfeld der Größe 3×3 betrachtet werden, welches iterativ über das Gitter „geschoben“ wird. Für jede Zelle im „aktiven Feld“ wird mittels einer Funktion getestet, ob die Zelle überlebt bzw. neu geboren wird (siehe 2.) und der Zustand der Zelle entsprechend aktualisiert.
2. Das (3×3) - Feld soll einer logischen Funktion ALIVE übergeben werden, welche anhand der Summe der Feldeinträge testet, ob der mittlere Feldeintrag überlebt bzw. geboren wird.
3. Schreibe eine Subroutine für die pseudo-grafische Ausgabe eines als Argument übergebenen Feldes beliebiger Größe. (Beachte, dass der Rand um das „aktive Feld“ nicht ausgegeben werden soll.)

```

+---+---+---+---+---+
|   |   |   |   |   |
+---+---+---+---+---+
|   |   |   |   |   |
+---+---+---+---+---+
|   |   |   | 0 |   |
+---+---+---+---+---+
|   |   |   | 0 |   |
+---+---+---+---+---+
|   | 0 | 0 | 0 |   |
+---+---+---+---+---+
|   |   |   |   |   |
+---+---+---+---+---+

```

4. Im Hauptprogramm soll zunächst ein INTEGER-Feld der Größe $(n+2) \times (n+2)$ allokiert werden, wobei n die Größe des Gitters bezeichne. Die Population lebt im inneren Teil des Feldes, der äußere Rand wird uns die Arbeit mit dem Feld erleichtern. Für die oben dargestellte Ausgangspopulation sollen anschließend mehrere Zeitschritte simuliert und jeweils das aktualisierte Gitter pseudo-grafisch ausgegeben werden.
5. **ZUSATZ:** Kompakt schreiben wir B3/S23 und meinen damit, dass eine leere Zelle mit genau drei Nachbarn besetzt wird („born“), und eine besetzte Zelle mit zwei oder drei Nachbarn überlebt („survives“), andernfalls stirbt die Zelle aus. Die oben beschriebenen Originalregeln können beliebig modifiziert werden.
 Schreibe eine weitere Funktion MODIFIED_ALIVE, welche neben dem (3×3) -Feld zwei eindimensionale INTEGER-Felder B und S als zusätzliche Parameter übergeben bekommt und die entsprechenden Regeln implementiert. Vergleiche nun die zeitliche Sequenz des obigen Musters für die Originalregeln B3/S23 und die „Highlife“-Regeln B6/S23.
6. **ZUSATZ:** Neben dem „Abscannen“ des Gitters mit einem 3×3 Raster gibt es eine weitere Möglichkeit, um die Anzahl lebender Nachbarn einer Zelle zu ermitteln: Wir verwenden ein zweites Feld gleicher Gestalt, dass wir zunächst mit Nullen initialisieren. Indem wir die Funktion CSHIFT geschachtelt verwenden, können wir nacheinander alle benachbarten Einträge einer Zelle auf den Wert an der entsprechenden Stelle im Hilfsfeld addieren. Nachdem wir 9-mal (bzw. effektiv 8-mal) in die verschiedenen Himmelsrichtungen geschiftet haben, steht in jedem Eintrag des Hilfsfeldes, wie viele Nachbarn die entsprechende Zelle hat. Mithilfe eines WHERE-Konstrukts kann nun jeder Zelle (des Originalfeldes) der Wert 0 oder 1 zugewiesen werden. So wird das ganze Feld auf einmal bearbeitet und nicht Eintrag für Eintrag.