

# Programmieren C: Arrays, Zufallszahlen: Nagelbrett simulieren, Balkendiagramm anzeigen

## Klaus Kusche

Wir wollen den klassischen Versuch mit dem Nagelbrett simulieren:

- Man lässt k Kugeln der Reihe nach mittig von oben auf r Reihen pyramidenförmig angeordneter, in jeder Reihe gegeneinander versetzter Nägel fallen:  
Die 1. Reihe enthält einen Nagel, die zweite zwei, die letzte r Nägel.
- Unter der untersten Nagelreihe sind  $(r+1)$  Fächer: Eines links vom ersten Nagel, eines rechts vom letzten Nagel, und je eines zwischen zwei Nägeln.  
Man zählt, wie viele Kugeln in jedem der  $(r+1)$  Fächer landen, wobei die Chance in jeder Reihe 50:50 ist, dass die Kugel links oder rechts am Nagel vorbeifällt.

### Hinweise:

- Das Programm wird mit 2 Zahlen r (Anzahl der Reihen) und k (Anzahl der Kugeln) aufgerufen.
- Lösungsidee:
  - Unser Array zählt die Kugeln pro Fach.  
Es muss also so viele Elemente haben, wie es Fächer gibt  
(also eines mehr als die eingegebene Anzahl der Reihen).
  - Ganz am Anfang sind alle Fächer leer (d.h. alle Zähler 0),  
ganz am Ende geben wir die Zählerstände aller Fächer aus.
  - Dazwischen simulieren wir die k Kugeln einzeln nacheinander,  
d.h. wir lassen in einer Schleife k Mal eine einzelne Kugel fallen.
    - Jede Kugel müssen wir r Mal nacheinander auf einen Nagel fallen lassen,  
d.h. r Mal zufällig 0 ("fällt nach links") oder 1 ("fällt nach rechts")  
zusammenzählen, wodurch sich eine Zahl anzahl\_rechts zwischen  
0 und r ergibt ("die Kugel ist anzahl\_rechts Mal nach rechts gefallen").  
  
Dabei ist die Reihenfolge der 0 und 1 egal,  
es kommt nur auf die Anzahl der Einser an:  
Alle Wege mit x Mal links und y Mal rechts landen im selben Fach,  
egal, ob die Kugel zuerst nach links und dann nach rechts fällt  
oder umgekehrt.
    - Dieses anzahl\_rechts ist die Nummer des Faches,  
dessen Zählerstand wir für diese Kugel um 1 erhöhen müssen.
  - Denk daran, dass dein Programm bei jedem Programmlauf andere Zufallszahlen berechnen soll!

So wirklich schön ist die Simulation erst mit einem Balkendiagramm als Ausgabe.  
Stelle daher die Verteilung der Kugeln (bzw. die Anzahl der Kugeln in jedem Fach)  
mit einem **waagrechten Balkendiagramm** dar  
(ein senkrechtes Balkendiagramm ist zwar schöner, aber viel schwieriger,  
im Übungsbeispiel "Noten zählen" ist die Idee dazu grob beschrieben).

Pro Fach soll eine Zeile ausgegeben werden:

- Am Anfang der Zeile soll die Anzahl der Kugeln in diesem Fach als Zahl stehen.
- Dann sollen je nach Anzahl der Kugeln mehr oder weniger '#' ausgegeben werden.

Am einfachsten ist es, genau so viele '#' auszugeben, wie es Kugeln im Fach gibt (d.h. beispielsweise 10 '#' für 10 Kugeln im Fach). Probiere das als Erstes.

Du wirst die '#' einzeln (eins nach dem anderen, bis genug ausgegeben sind) ausgeben müssen, es gibt keinen fertigen Befehl, um eine bestimmte Anzahl '#' auszugeben.

Das ist aber keine schöne Lösung: Für Versuche mit wenigen Kugeln sind die Balken winzig kurz, für ein Diagramm mit tausenden Kugeln viel zu lang (länger als die Zeile).

Wir wollen also unsere Balken so zeichnen,  
dass der Balken für das Fach mit der größten Anzahl bis 1 Zeichen vor das Zeilenende reicht  
(egal, ob die größte Anzahl von Kugeln in einem Fach jetzt 3 oder 300 ist)  
und alle anderen Balken entsprechend kürzer sind.

Verwende dazu folgende Idee:

- Gehe vor der Ausgabe alle Zahlen im Array einzeln der Reihe nach durch und ermittle (so wie in einer früheren Übung) deren Maximum.
- Berechne dann aus diesem Maximum und der Zeilenlänge (abzüglich dem Platz, den die Ausgabe der Anzahl als Zahl braucht) den Faktor, mit dem du die Kugel-Anzahl multiplizieren musst, damit du auf die Anzahl der '#' kommst.  
Für die Zahl, die das Maximum ist, muss bei der Multiplikation mit diesem Faktor genau die größtmögliche Balkenlänge herauskommen, für alle anderen Zahlen proportional weniger.

Um Probleme mit dem Abschneiden des **int**-Divisionsergebnisses zu vermeiden, wird diese Rechnung und der Faktor wohl **double** sein.

- Gehe als letzten Schritt noch einmal alle Fächer einzeln durch und gib die Anzahl gefolgt von "Faktor mal Anzahl" vielen '#' aus.