Programmieren 1 Übung: Erste Arrays

Klaus Kusche

1.) Noten zählen

Schreib ein Programm, das mit mehreren Noten (*ganzen* Zahlen zwischen 1 und 6)
auf der Befehlszeile aufgerufen wird und als Liste ausgibt, **wie oft jede Note vorkommt**.

Hinweise:

- Du sollst nicht sechs einzelne Zählvariablen verwenden, um zu zählen,
  wie oft jede Note vorkommt, sondern ein *Array* mit sechs Elementen
  (wenn du willst, darfst du auch ein Array mit *sieben* Elementen verwenden
  und das nullte Element leer lassen, dann ist der Index gleich der Note).
- Du wirst in deinem Programm *drei Schleifen* brauchen:
  - Eine, um am Anfang die Zähler der Reihe nach alle auf 0 zu setzen.
  - Die zweite, um dann eine Note nach der anderen von der Befehlszeile
    einzulesen und den entsprechenden Zähler um eins zu erhöhen.
  - Und am Schluss eine dritte, um für jeden Notenwert von 1 bis 6
    zeilenweise die Note und die dazugehörige Anzahl auszugeben.
- Das Programm soll erkennen (eine *Fehlermeldung* ausgeben und enden),
  wenn es mit *gar keinen Noten* aufgerufen wird,
  oder wenn eine *Zahl kleiner 1 oder größer 6* eingegeben worden ist
  (wie prüft man *zwei Bedingungen* in einem **`if`** ?).

Zusatzaufgaben:

1. Baue das Programm so um, dass nicht Noten,
   sondern **Punkte (von hundert) bzw. Prozent eingegeben** werden.

   Die Noten berechnen sich im Berufskolleg aus den Punkten wie folgt:
   `lround((600 - 5 * punkte) / 100)`
   (**`lround`** ist die kaufmännische Rundung, die Funktion kommt aus **`math.h`**)

   Achtung: Die Formel oben stimmt zwar aus mathematischer Sicht,
   aber damit sie in C richtig rechnet, *muss man noch etwas ändern*!!!

   Auch bei den eingegebenen Punkten soll *geprüft* werden,
   ob sie *zwischen 0 und 100* liegen.
2. Gib ganz am Ende des Programms auch den **Mittelwert der Punkte**
   (als Kommazahl!) aus.
3. Gib in der Liste bei jeder Note nicht nur aus, wie oft sie vorgekommen ist,
   sondern auch, welchen **prozentuellen Anteil am Gesamtergebnis** sie hat
   (d.h. wie viele Prozent der Schüler haben diese Note), und zwar *mit Kommastellen*.

2.) Balkendiagramme

Fortsetzung der *vorigen Übung*:

Stelle die Verteilung der Noten mit einem **waagrechten Balkendiagramm** dar.

Pro Note soll eine Zeile ausgegeben werden:

- Am Anfang der Zeile soll die Note, ihre Anzahl und ihr Prozent-Anteil stehen, schön in Spalten mit fixer Breite.
- Dann sollen je nach Anzahl der Schüler mit dieser Note mehr oder weniger '**#**' ausgegeben werden.

Am einfachsten ist es, genau so viele '**#**' auszugeben, wie es Schüler mit dieser Note gibt (d.h. beispielsweise 10 '**#**' für 10 Schüler mit dieser Note). 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 kleine Klassen sind die Balken winzig kurz, für ein Noten-Diagramm der ganzen Schule viel zu lang (länger als die Zeile).

Wir wollen also unsere Balken so zeichnen,
dass der *Balken für die Note mit der größten Anzahl bis 1 Zeichen vor das Zeilenende reicht*
(egal, ob die größte Anzahl von Schülern mit derselben Note 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 von Note, Anzahl und Anteil braucht) den *Faktor*, mit dem du die Schüler-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 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 Noten einzeln durch und gib die Note samt Anzahl und Anteil gefolgt von "Faktor mal Anzahl" vielen '**#**' aus.

Zusatzaufgabe für Tüftler (schwierig!!):

Schaffst du auch ein **senkrechtes Balkendiagramm**?

Du musst das Diagramm *zeilenweise* ausgeben und in jeder Zeile alle Noten durchgehen.
Ist der Balken der Note mindestens so hoch wie die aktuelle Zeile,
musst du ein Stückchen Balken ausgeben, sonst ein Stückchen "nichts".

Berechne dazu wieder das Maximum aller Anzahlen.
Wenn das Balkendiagramm eine *fixe Höhe* von *z* Zeilen haben soll,
lässt sich daraus berechnen, welche Anzahl eine Höhe von einer Zeile darstellt.

3.) Simulation des Nagelbrettes

Wenn du das vorige Beispiel mit den Noten-Balkendiagrammen gelöst hast,
wollen wir das Programm gleich zur Lösung eines ganz ähnlichen Problems umbauen:
Wir zählen Kugeln statt Noten und stellen das Ergebnis wieder mit Balken dar.

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 jetzt nicht mehr Noten, sondern 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.
- So wirklich schön ist die Simulation erst mit einem *Balkendiagramm* als Ausgabe,
  also geben wir so wie im vorigen Beispiel wieder waagrechte Balken aus.

  Passe deine Balken an Bildschirmbreite und Ergebnis an: Die Skalierung
  sollte so sein, dass der längste Balken gerade die Bildschirmbreite füllt.
- Denk daran, dass dein Programm bei jedem Programmlauf *andere Zufallszahlen*
  berechnen soll!