Programmieren für Physiker

Interfakultatives Institut für Anwendungen der Informatik
Institut für Theoretische Teilchenphysik

Prof. Dr. M. Steinhauser, Dr. A. Mildenberger
`http://comp.physik.kit.edu`

SS 2017 – Blatt 08
Bearbeitungszeitraum: bis 21. Juni 2017

Aufgabe 19: Erzeugung aller Permutationen **Pflichtaufgabe**

In dieser Aufgabe soll ein Verfahren programmiert werden, welches in effektiver Weise sämtliche Permutationen erzeugt. Die Permutationen werden in *lexikographischer* Reihenfolge generiert, d.h. in aufsteigender Reihenfolge nach stellenweisem Vergleich mit $\leq$. Um alle Permutationen zu erhalten, muss deswegen mit einer geordneten Folge

$$a_1 \leq a_2 \leq \ldots \leq a_n$$

begonnen werden. (Die Werte müssen hierbei nicht notwendigerweise paarweise verschieden sein.)
Der hier angegebene Algorithmus, um jeweils die nächste Permutation zu erzeugen, ist seit mindestens dem 14. Jahrhundert in Indien bekannt und wurde seither mehrfach „wiederentdeckt“.

- P1: Finde das größte k, so dass $a_k < a_{k+1}$ gilt. Falls dies nicht existiert, beende den Algorithmus, da die letztmögliche Permutation erzeugt wurde.
- P2: Finde das größte l, so dass $a_k < a_l$ gilt. Dieses muss nach dem ersten Schritt auf alle Fälle existieren.
- P3: Tausche a_k und a_l.
- P4: Drehe die Reihenfolge aller Elemente zwischen $k+1$ und n um, dabei sollen die Indizes $k+1$ und n mit eingeschlossen sein.
  Damit ist die nächste Permutation erzeugt und kann verarbeitet werden.

In dieser Aufgabe soll dieses Verfahren in einer Funktion mit Namen `next_permutation` programmiert werden. Die Funktion soll als Argument ein `char`-Feld und dessen Länge erhalten, der Rückgabewert soll ein `bool`-Wert sein, der angibt, ob eine neue Permutation erzeugt werden konnte. Falls ja, ist die Permutation dann im `char`-Feld erhalten.

In Punkt P3 und P4 des obigen Algorithmus sind jeweils Werte zu tauschen. Schreiben und verwenden Sie hierfür bitte eine geeignet programmierte *Swap*-Funktion mit Referenzparametern.

Bei dieser Aufgabe wird ein Hauptprogramm zur Verfügung gestellt, dieses ruft die Permutationsfunktion auf. Sie erhalten es von der Webseite oder aus dem Verzeichnis `/home/ck17/daten/`.

Zusatzfragen (freiwillig): 1. Überlegen Sie sich, wie Sie die Anzahl der Anzahl der Permutationen mathematisch ermitteln können, auch für den Fall, dass Zeichen mehrfach vorkommen.
2. Erklären Sie die Funktionsweise des Algorithmus.

Aufgabe 20: Rechteck-Struct **Pflichtaufgabe**

Es ist ein eigener Datentyp `rechteck` mittels `struct` zu schreiben. Die Rechtecke sind dabei durch zwei Gleitkommazahlen, also Breite und Höhe charakterisiert.
Programmieren Sie bitte die beiden folgenden (freien) Funktionen, es wird noch keine Objekorientierung erwartet:

(a) Funktion `rotiere` mit einem `rechteck` als Argument und als Rüchgabewert. Letzteres soll um 90° rotiert sein.
(b) Funktion `bedeckt` mit zwei Rechteckargumenten und einem Logikfunktionswert. Es soll getestet werden, ob das erste Rechteck das zweite komplett bedecken kann. Dabei genügt es, die Rechtecke in der gegebenen Orientierung zu betrachen und lediglich die Höhen und Breiten miteinander zu vergleichen.

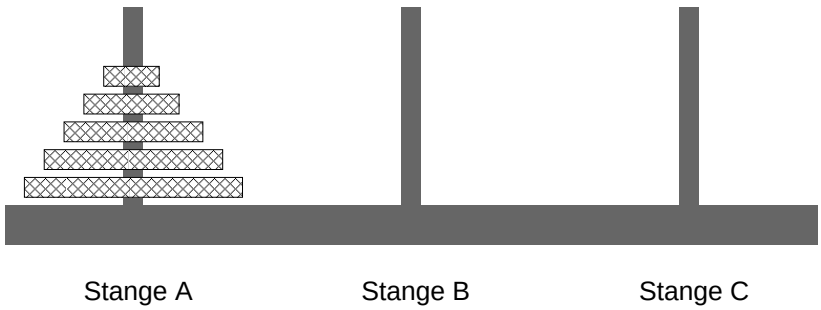
Ein Hauptprogramm, welches diese Funktionen aufruft, ist auf der Webseite oder im Verzeichnis `/home/ck17/daten/` zu finden.

Aufgabe 21: Quersumme **freiwillig**

Schreiben Sie eine rekursive Funktion, die die Quersumme, also die Summe der Ziffern, einer positiven Zahl berechnet.
Verwenden Sie diese Funktion, um eine zweite rekursive Funktion zu erstellen, die die vollständig reduzierte Quersumme berechnet. Letztere entsteht durch wiederholtes Bilden der Quersumme, bis das Ergebnis einstellig ist.
Lassen Sie den Benutzer im Hauptprogramm eine Zahl eingeben und geben Sie die Quersumme und die vollständig reduzierte Quersumme aus.

Aufgabe 22: Türme von Hanoi **freiwillig**

Das Spiel *Türme von Hanoi* besteht aus n verschieden großen Scheiben und drei Stangen. Zu Beginn liegen alle Scheiben, der Größe nach sortiert, auf der ersten Stange. Ziel des Spieles ist es, alle Scheiben von der ersten auf eine andere Stange zu versetzen. Dabei sind jedoch folgende Regeln zu beachten: Es darf immer nur eine Scheibe zu einem Zeitpunkt bewegt werden und es darf nie eine größere auf eine kleinere Scheibe gelegt werden. Eine Skizze der Ausgangsposition mit 5 Scheiben:



Um ein Gefühl für die Problemstellung zu erhalten, ist es ratsam, sich zunächst die Lösung für $n = 1$, 2, 3 und evtl. $n = 4$ zu überlegen.
Ist Ihnen aufgefallen, dass Sie das Vorgehen für n Scheiben auf die Lösung für $n - 1$ Scheiben zurückführen können? Nehmen Sie an, Sie hätten ein Verfahren, um die obersten $n - 1$ Scheiben zu versetzen. Welche Schritte sind dann nötig, um den Turm der Größe n zu versetzen? Dieser Lösungsweg kann mittels Rekursion in einem Programm umgesetzt werden.
Entwickeln Sie ein Programm, das für ein einzugebendes n alle einzelnen Scheibenbewegungen ausgibt, die nötig sind, um das Problem zu lösen.
