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 2023 – Blatt 07
Bearbeitung bis 14. Juni 2023

Aufgabe 17: Erzeugung aller Permutationen **Pflichtaufgabe**

In dieser Aufgabe soll ein Verfahren programmiert werden, das in effektiver Weise sämtliche Permutationen von n Werten a_i 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 brauchen nicht notwendigerweise paarweise verschieden zu 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.

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 18: Minuten-Sekunden-Struct **Pflichtaufgabe**

Schreiben Sie mittels `struct` einen Datentyp `minsek`, der zur Aufbewahrung von Zeiten mit Minuten- und Sekundenwert dient und dazu zwei `int`-Variablen verwendet.
Programmieren Sie bitte folgende drei Funktionen, die jeweils `minsek` Argumente oder Rückgabewerte verwenden: (a) Eingeben einer Zeit, dabei sollen Minuten und Sekunde eingegeben werden. (b) Die Ausgabe in Form Minuten:Sekunden eines derartigen Structs. (c) Die Addition von zwei Zeiten, das Ergebnis soll wieder vom Typ `minsek` sein. Berücksichtigen Sie dabei bitte eventuelle Überträge der Sekundenvariable zu den Minuten, so dass die Sekundenvariable stets im Intervall $[0, 59]$ ist.

Rufen Sie im Hauptprogramm diese Funktionen so auf, dass Ihr Programm zwei eingegebene Zeiten addiert.

Zusatz, freiwillig: Überladen Sie den Operator + so, dass damit die Addition von zwei `minsek`-structs möglich ist.

Aufgabe 19: Wie oft durch zwei teilbar? **freiwillig**

Schreiben Sie eine rekursive Funktion, die ermittelt, wie oft eine ganze Zahl durch zwei teilbar ist. Vom Hauptprogramm aus soll die Funktion mit eingegebenen Zahlen aufgerufen werden und das Ergebnis ausgegeben werden.

Tipp: Rekursiver Aufruf mit $x/2$, Rückgabewert durch Aufaddition.

Erweiterung (freiwillig): Ermitteln Sie auf rekursive Weise die Anzahl der Primfaktoren.
