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 09
Bearbeitungszeitraum: bis 28. Juni 2017

Aufgabe 23: Quicksort **Pflichtaufgabe**

Quicksort (C.A.R. Hoare, 1961/62) ist ein schneller, rekursiver Sortieralgorithmus. Hier eine Skizze der Funktionsweise: Es wird aus der zu sortierenden Liste ein „Pivotelement“ m gewählt (im einfachsten Fall kann das erste Element der Liste benutzt werden) und die Liste wird in einem Durchgang in zwei Teillisten geordnet. In der linken Teilliste sind alle Elemente $\leq m$, alle Elemente der rechten Teilliste sind $\geq m$. Dieser Schritt heißt auch Partitionierung. Nun können die beiden noch unsortierten Teillisten jeweils für sich mit dem gleichen Verfahren bearbeitet werden, dies geschieht rekursiv. Teillisten mit weniger als zwei Elementen brauchen natürlich nicht mehr sortiert zu werden.
Schreiben Sie eine Funktion `QuickSort (double A[], int s, int t)`, die im Feld `A` zwischen den Indizes `s` und `t` sortiert. Bei Bedarf soll diese Funktion eine Partitionierung durchführen und sich selbst aufrufen. Entwerfen Sie eine eigene Partitionierung oder verwenden Sie folgendes Schema:

```
Partitionierung (A, s, t)
  pivot := A[s]
  l := s
  Schleife i von s+1 bis t
    Falls A[i] < pivot
    dann:  l := l+1
           Tausche (A[i],A[l])
  Tausche (A[s],A[l])
```

Wie funktioniert diese Partitionierung? Wo steht nach Ausführung dieser Routine das Pivotelement, welche beiden Teillisten sind nun also zu sortieren?

Um Ihre Sortierfunktion zu testen, generieren Sie ein Feld mit 1000 zufälligen `double`-Werten im Bereich $[0, 1]$.

Die Zufallszahlen können Sie folgendermaßen erzeugen: Mit den Anweisungen `#include <cstdlib>` und `#include <ctime>` im Kopf des Programms ergibt `double(rand())/RAND_MAX` bei jedem Aufruf eine gleichverteilte Zufallszahl im Intervall $[0, 1[$. Um den Generator zu Beginn auf einen zufälligen Startwert zu setzen, benötigen Sie noch ein einmaliges `srand((unsigned int)time(0))` am Anfang des Programmcodes.

Verwenden Sie zum Tausch von Feldelementen und für die Partitionierung jeweils Funktionen.

Überlegen Sie sich bitte, wie viele und welche Modifikationen in Ihrem Programm nötig sind, um die Zahlenliste absteigend statt aufsteigend zu sortieren.

Zusatzfragen (freiwillig): In welchen Ausgangssituationen ist die Laufzeit von Quicksort in der obigen Version am ungünstigsten, d.h. sind am meisten Vergleiche durchzuführen? Messen Sie die Laufzeit des Programms mit geeignet langen Zahlenfeldern für verschiedene Ausgangssituationen.

Aufgabe 24: rationale Zahlen **Pflichtaufgabe**

Programmieren Sie in C++ eine Klasse `class Ratio {...}`, die das Rechnen mit Brüchen ermöglicht. Zähler und Nenner sollen von außen unzugängliche `long` Variablen sein. Implementieren Sie folgende Methoden:

- Eine Wertzuweisungsfunktion mit zwei Argumenten, um Zähler und Nenner einen Wert zuzuweisen. (Falls Ihnen bereits Konstruktoren vertraut sind, können auch diese verwendet werden.)
- Kürzen (mittels Berechnung des größten gemeinsamen Teilers, siehe hierzu entweder Vorlesung oder Euklidischer Algorithmus).
- Addition zweier Brüche
- Subtraktion zweier Brüche
- Multiplikation zweier Brüche
- Division zweier Brüche
- Unäres Minus (d.h. Minus als Vorzeichen)
- Ausgabe in der Form *Zähler/Nenner*
- Berechnung des Gleitkommawerts (`double`) des Bruchs

Bei allen Rechenoperationen soll nach der eigentlichen Berechnung das Ergebnis sofort vollständig gekürzt werden. Um die Rechenoperationen bequem aufrufen zu können, überladen Sie bitte die Operatoren `+ - * /`.
Achten Sie darauf, dass auch die Rechnung mit negativen Brüchen richtig funktioniert.
Berechnen Sie damit im Hauptprogramm

$$\frac{2}{25}\Big/\frac{7}{5}-\frac{2}{5}\cdot\left(-\frac{1}{4}+\frac{1}{3}\right)$$

und geben Sie den Wert als Bruch und als Gleitkommazahl aus. Die einzelnen Brüche dürfen hierbei explizit im Hauptprogramm eingetragen sein.

Zum Knobeln: Zahlen in Summanden zerlegen **freiwillig**

Die Zahl 5 kann auf genau 7 verschiedene Arten als Summe von positiven ganzen Zahlen geschrieben werden:

$$(1+1+1+1+1),\ (1+1+1+2),\ (1+2+2),\ (1+1+3),\ (2+3),\ (1+4),\ (5).$$

Die Reihenfolge der Summanden spielt keine Rolle.
Allgemein bezeichne $p(n)$, $n \in \mathbb{N}$ die Anzahl der verschiedenen Möglichkeiten, die Zahl n als Summe von positiven ganzen Zahlen zu schreiben. Es ist also $p(5) = 7$.
Entwickeln und programmieren Sie ein Verfahren, um $p(n)$ für ein gegebenes n zu berechnen.
Einige weitere Werte zum Testen: $p(12) = 77$, $p(34) = 12310$, $p(123) = 2552338241$.
