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 09
Bearbeitung bis 28. Juni 2023

Bitte denken Sie daran, sich für die Klausur am 24.07.23 im Studierendenportal anzumelden. Zur Teilnahme sind 80% der Pflichtaufgaben erfolgreich zu lösen, diese Vorleistung wird nicht im Studierendenportal verbucht.

Aufgabe 23: 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 per Konstruktor mit zwei Argumenten, um Zähler und Nenner zu setzen.
  Falls benötigt, können Sie gerne weitere Konstruktoren schreiben.
- Kürzen (mittels Berechnung des größten gemeinsamen Teilers, siehe hierzu entweder Vorlesung oder recherchieren Sie den Euklidischen Algorithmus).
- Addition zweier Brüche
- Subtraktion zweier Brüche
- Multiplikation zweier Brüche
- Division zweier Brüche
- Unäres Minus (d.h. Minus als Vorzeichen)
- Ausgaberoutine (ohne Argument) in der Form *Zähler/Nenner*.
  Gerne können Sie den `operator<<` verwenden.
- 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

$$\frac{2}{15}\Big/\frac{7}{5}+\frac{3}{7}\cdot\left(-\frac{1}{2}+\frac{1}{3}\right)$$

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

Aufgabe 24: Bordüre **freiwilliges Zusatztestat**[1]

Ihr Freund Fritz ist Fliesenleger, der gerne Bordüren legt, also eindimensionale Ketten von gemusterten Fliesen. Er hat dafür genau drei verschiedene Typen von Fliesen zur Verfügung:  ,  und  . Wie abgebildet, haben diese Bausteine verschiedene Längen: eine, zwei oder drei Einheiten. Damit soll nun eine Kette der Länge n Einheiten gebildet werden, allerdings dürfen dabei niemals zwei identische Muster direkt aneinander liegen. Auch dürfen die Fliesen nicht geteilt oder geschnitten werden.
Fritz fragt sich, ob man wohl ermitteln könne, wie viele Möglichkeiten $m(n)$ es unter den gegebenen Bedingungen gibt, eine Kette der Länge n zu legen. Helfen Sie ihm, indem Sie ein geeignetes Programm schreiben und eine Tabelle von $m(n)$ für $1 \leq n \leq 30$ ausgeben.

Beispiel: m(5) = 4, da es genau die folgenden vier Möglichkeiten gibt:


Tipp: Rekursive Funktion, die als Argument die Restlänge und den zuletzt verwendeten Baustein enthält. Als Rückgabewert kann die Anzahl der gefundenen Möglichkeiten für diese Ausgangskonstellation verwendet werden.
Zwei Kontrollwerte zum Testen: $m(10) = 27$, $m(15) = 164$.

Aufgabe 25: Auflösung einer Rekursion **freiwillig**

Gegeben sei eine rekursive Funktionen, die folgende, relativ einfache Struktur habe:

```
Type2 f (Type1 x)
{
  if( test(x)) return q(x) ;
  return f(p(x)) ;
}
```

Dabei sind `Type1 p (Type1)` und `Type2 q (Type1)` sowie `bool test (Type1)` beliebige Funktionen, die hier für Rechenoperationen mit dem Argument `x` stehen. Geben Sie mit Hilfe einer Schleife eine nicht-rekursive Formulierung der obigen Funktion `f` an.

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: $p(12) = 77$, $p(34) = 12310$, $p(123) = 2552338241$.

---

[1] Sie können bei dieser freiwilligen Aufgabe ein Testat erhalten, welches gewertet wird; die Aufgabe zählt aber nicht als Pflichtaufgabe.