Dirk Praetorius,
Michele Ruggeri

Sommersemester 2017
30.05.2017

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 11

Aufgabe 11.1. Schreiben Sie eine Klasse `Fraction` zur Darstellung eines Bruchs $x = p/q$, wobei $p \in \mathbb{Z}$ und $q \in \mathbb{N}$ als `int` gespeichert werden. Implementieren Sie

- den Standardkonstruktor (ohne Parameter), der $p = 0$ und $q = 1$ setzt,
- einen Konstruktor, der $p, q \in \mathbb{Z}$ mit $q \neq 0$ als Input übernimmt und den Bruch speichert,
- den Kopierkonstruktor,
- den Zuweisungsoperator und
- den Destruktor.

Stellen Sie mittels `assert` sicher, dass die Übergabeparameter zulässig sind, d.h. $q \neq 0$. Beachten Sie den Fall $q < 0$, bei dem intern $(-p)/|q|$ gespeichert wird. Implementieren Sie darüber hinaus die Zugriffsmethoden

- `setNumerator`, `getNumerator` für den Zähler und
- `setDenominator`, `getDenominator` für den Nenner.

Testen Sie ihre Implementierung geeignet! Speichern Sie den Source-Code unter `fraction.{hpp/cpp}` in das Verzeichnis `serie11`.

Aufgabe 11.2. Implementieren Sie eine Methode `reduce` für die Klasse `Fraction` aus Aufgabe 11.1, um einen Bruch auf die gekürzte Form $x = p/q$ zu bringen (mit $p \in \mathbb{Z}$ und $q \in \mathbb{N}$ teilerfremd). Verwenden Sie einen beliebigen Algorithmus, um den größten gemeinsamen Teiler von Nenner und Teiler zu bestimmen, z.B. den Euklid-Algorithmus (VO Folien 96–98 und Folie 106) oder einen Algorithmus für die Primfaktorzerlegung von Integern (Aufgabe 9.2 und Aufgabe 9.3). Testen Sie Ihre Implementierung geeignet!

Aufgabe 11.3. Implementieren Sie das Type Casting von `Fraction` aus Aufgabe 11.1 auf `double`. Implementieren Sie auch das Type Casting von double auf `Fraction`. Berücksichtigen Sie nur die ersten 9 Nachkommastellen (*Hinweis:* Übersetzen Sie diesen Anteil in einen Bruch mit dem Nenner 10^9 und kürzen Sie dann). Implementieren Sie ferner das Type Casting von `Fraction` auf `int`, das einen Bruch auf die nächste Ganzzahl rundet. Beträgt der Nachkommaanteil im mathematischen Sinn genau 0.5 so unterscheiden Sie die folgenden beiden Fälle: Ist der Bruch positiv, so runden Sie auf; d.h. aus $3/2$ wird 2. Ist der Bruch negativ, so runden Sie ab; d.h. aus $-3/2$ wird -2 Testen Sie Ihre Implementierung geeignet!

Aufgabe 11.4. Überladen Sie die Operatoren $+$, $-$, $*$ und $/$ um die Summe, die Differenz, das Produkt und den Quotienten zweier Brüche im Format `Fraction` aus Aufgabe 11.1 zu berechnen. Stellen Sie bei $/$ mittels `assert` sicher, dass Sie nicht durch 0 dividieren. Das Ergebnis soll in allen Fällen gekürzte Form haben. Testen Sie ihre Implementierung geeignet!

Aufgabe 11.5. Überladen Sie den Vorzeichenoperator $-$, der zum `Fraction` x den `Fraction` $-x$ liefert, und den `<<`-Operator, um einen `Fraction` $x := p/q$ in der Form `p/q` ausgeben zu können (siehe Folie 284 für ein Beispiel mit der Klasse `Complex` aus der Vorlesung). Testen Sie Ihre Implementierung geeignet!

Aufgabe 11.6. Überladen Sie die Vergleichsoperatoren `==`, `!=`, `<`, `<=`, `>`, `>=` für die Klasse `Fraction` aus Aufgabe 11.1. Vergleichen dabei Sie Zähler und Nenner der Brüche direkt (*Hinweis:* Sie müssen einen gemeinsamen Nenner bestimmen) und verwenden Sie keines der Type Casts aus Aufgabe 11.3 Testen Sie Ihre Implementierung geeignet!

Aufgabe 11.7. Schreiben Sie eine Klasse `FractionVector` zur Speicherung von Vektoren in $\mathbb{Q}^n$. Hierbei sollen die Länge n (`int`) sowie der Koeffizientenvektor (`Fraction*`) abgespeichert werden. Implementieren Sie

- den Konstruktor,
- den Kopierkonstruktor,
- den Zuweisungsoperator und
- den Destruktor.

Implementieren Sie darüber hinaus die Zugriffsmethoden

- `setCoefficient`, `getCoefficient` für die Vektoreinträge und
- `getLength` für die Länge.

Testen Sie Ihre Implementierung geeignet! Speichern Sie den Source-Code unter `fractionVector.{hpp/cpp}` in das Verzeichnis `serie11`.

Aufgabe 11.8. Schreiben Sie die Methode `sort` für die Klasse `FractionVector` aus Aufgabe 11.7, die einen Vektor aufsteigend sortiert und mit dem sortierten Koeffizientenvektor überschreibt. Der Vektor

$$x = \left(-\frac{1}{2}, \frac{5}{7}, \frac{1}{3}, \frac{0}{1}, \frac{11}{2}, -\frac{7}{8}\right) \in \mathbb{Q}^6$$

soll beispielweise durch Aufruf von `sort` durch

$$x = \left(-\frac{7}{8}, -\frac{1}{2}, \frac{0}{1}, \frac{1}{3}, \frac{5}{7}, \frac{11}{2}\right)$$

überschrieben werden. Verwenden Sie die Vergleichsoperatoren aus Aufgabe 11.6 und einen beliebigen Sortieralgorithmus. Wie groß ist der Aufwand Ihrer Implementierung? Testen Sie Ihren Code geeignet!