

Übung 8

Klassen & Operatoren

A 8.1: Klasse Vektor (Teil 2.)

(3)

Überladen Sie die Operatoren `=`, `+`, `*` der Klasse `Vektor` (siehe die Aufgabe A 7.2). Die Ergebnisse der Operationen `+` und `add` als auch `*` und `crossMul` sollten übereinstimmen.

Testen Sie die Klasse mit dem folgenden Code-Fragment:

```
Vektor v1 = Vektor(17.4, 0.0, 0.0);  
Vektor v2 = Vektor( 0.0, 4.8, 0.0);  
Vektor v3 = v1.crossMul(v2);  
Vektor v4 = v1;  
Vektor v5 = v1+v2;  
Vektor v6 = v1+v2*v3;
```

A 8.2: Klasse Bruch

(12)

- (a) Deklarieren und definieren Sie eine C++ - Klasse `Bruch`, die Brüche und Operationen auf diese kapselt. Ein `Bruch` soll dabei durch Angabe von *Zähler* und *Nenner* konstruiert werden. Fangen Sie eventuelle Fehler bei der Initialisierung (z.B. `Nenner == 0`) ab.
- (b) Erweitern Sie die Klasse um eine Methode `gebeAus()`, die einen `Bruch` in einer der folgenden Formen ausgibt.
"Zähler / Nenner " (also z.B. **3 / 2**)
"double(Zähler)/double(Nenner)" (also **1.5**)
Das Ausgabeformat ist durch eine Variable `int static format` zu setzen.
- (c) Überladen Sie die Operatoren `=`, `+`, `-`, `*`, und `/`. Der `+` und der `-` Operator sollen dabei in ihrer *unären* Variante überladen werden.
- (d) Überladen Sie die Operatoren `+=`, `-=`, `*=` und `/=`.
- (e) Erweitern Sie die Klasse um die Möglichkeit, die Operationen `=`, `+`, `-`, `*`, `+=`, `-=`, `*=` und `/=` auch für beliebige Kombinationen von Brüchen und `double`-Zahlen zu verwenden.

Testen Sie die Funktionalität der Klasse anhand der folgenden Beispiele. Dabei soll sich eine Veränderung von **a**, **b** oder **c** nicht fortpflanzen, d.h. wenn **c** z.B. bei **c*=(b-a)** ein neuer Wert zugewiesen wird, soll im nächsten Teil wieder **c = 17 / 4** gelten!

$$a = \frac{5}{7} \quad b = \frac{3}{2} \quad c = \frac{17}{4}$$

$d = (a + b) * c / (a - b)$	$c * = (b - a)$	$f / = (a * b * c)$
$g = a + 7$	$h = b / 4$	$a * = 5$

A 8.3 Liste von "Adressen" 2.

(8)

Definieren Sie eine doppeltverkettete Liste von mindestens 3 Elementen vom Typ Adresse (siehe Aufg. 4.1) und ermöglichen Sie dem Benutzer, eine neue Adresse einzugeben (siehe Aufg. 5.1). Im Gegensatz zur Aufgabe 5.1 sollte dies mittels einer C++ Klasse `ListenKlasse` erfolgen, die u.A. die Methoden `Ausgabe` und `Sortiere` (siehe Unten) enthält. Die einzelnen Listenelemente sind durch eine Klasse `ListenElement` zu repräsentieren, wobei der Operator `<` in folgender Weise zu überladen ist: Seien A und B zwei Elemente der Klasse `ListenElement`. Dann ist A *kleiner als* B, d.h.

$(A < B) == 1$, wenn der Name-, Vorname- oder Ort- Eintrag von A dem entsprechenden Eintrag von B *alphabetisch untergeordnet* ist (lexikographisch). Eine static Variable `SortierArt` der Klasse `ListenElement` sollte bestimmen, durch welchen der drei Einträge der Vergleich $A < B$ bestimmt ist. Beispiel: Ist die Variable `SortierArt == 1`, dann soll die Liste nach Namen lexikographisch geordnet werden, etc. Die Methode `Sortiere` sollte die Listenelemente ausschließlich mittels Pointer und der überladenen Operation `<` sortieren.