Hochschule Worms
University of Applied Sciences
Fachbereich Informatik
Prof. Dr. Bernd Ruhland

Objektorentierte Programmierung

Übungsaufgabe

Operatoren zur Klasse der komplexen Brüche

Lernziele:

- Aggregation von Klassen
- Wiederverwendung von Code
- Operatoren
- Strukturiertes planvolles schrittweises Vorgehen
- Aufgabenstellung genau durchlesen

Aufgabenstellung:

Erstellen Sie eine Klasse ***cKompRat***, die es ermöglicht, komplexe rationale Zahlen zu verwalten und mit ihnen zu rechnen. Komplexe rationale Zahlen bestehen aus zwei rationalen Zahlen, also aus zwei Brüchen.

Die Klasse ***cKompRat*** hat als Bestandteile zwei Instanzen der bereits bekannten Klasse ***cBruch***, die es ermöglicht, Brüche zu verwalten und mit ihnen zu rechnen. Eine der beiden ***cBruch***-Instanzen nimmt den reelen Teil der komplexen Zahl auf, die andere den imaginären Teil.

Implementieren Sie folgende Funktionalitäten:

- Einen universellen Konstruktor, der die komplex rationale Zahl mit 0 initialisiert, falls keine Parameter angegeben sind (Vorgabewerte).

- Eine Vergleichsfunktion ***int kompRatVergleich (cKompRat lhs, cKompRat rhs)***, die folgende Werte zurückgibt:
  1 wenn lhs größer ist rhs
  0 wenn beide komplex rationalen Zahlen gleich groß sind
  -1 wenn lhs kleiner ist als rhs

- Einen Ausgabeoperator **<<** welcher die komplex rationale Zahl ausgibt, indem er die Ausgabe-Operatoren der Brüche für die beiden Bestandteile aufruft, die den Bruch in der Form Zähler/Nenner sowie als Gleitkommazahl ausgeben.

- Für die vier Grundrechenarten die folgenden Operatoren als Methoden (Mitgliedsfunktionen) der Klasse cKompRat:

  + - / *

  Was müssen Sie bei Methoden bezüglich der Anzahl der Parameter beachten?

- Folgende Vergleichsoperatoren vom Typ ***bool***:

  < == >

- Einen Präfix-Operator **~** bei dem der Real- und der Imaginärteil der komplex rationalen Zahl vertauscht werden. Rückgabewert der Funktion ist der Zustand **nach** der Vertauschung.

Verwenden Sie **Referenzen** überall wo es sinnvoll und möglich ist.

Schreiben Sie eine geeignete Testumgebung, mit der Sie die Funktionalitäten überprüfen können.

Ergänzungsaufgabe für die, die noch mehr mit den komplexen Brüchen spielen möchten:

- Erstellen Sie eine Funktion ***tausch()***, die zwei Instanzen von ***cKompRat*** miteinander vertauscht.
- Erstellen Sie eine Funktion ***sortier()***, die das Array der Brüche aufsteigend sortiert. Verwenden Sie den Bubblesort-Algorithmus.
- Definieren Sie im Hauptprogramm ein Array aus 8 ***cKompRat*** Objekten mit willkürlich gewählten Werten.
- Geben Sie das unsortierte Array aus.
- Sortieren Sie das Array mit den beiden Hilfsfunktionen.
- Geben Sie das sortierte Array nochmal aus.

Der Algorithmus des Bubblesort kann wie folgt umgesetzt werden:

```
// Bubble Sort Algorithmus (unoptimiert) bei ANZ Objekten

for (int n = ANZ; n>1; n--) {
      for (int i=0; i<n-1; i++) {
            if (elem[i] > elem[i+1]) {       // Elemente vergleichen
                  tausch(elem[i], elem[i+1]); // Elemente tauschen
            }
      }
}
```
