ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

cgl computer graphics lab

Department Informatik
Dr. Ronny Peikert

20. Januar 2004

37-847 Informatik — Übung 11 — WS 03/04

Am 27. Januar findet eine Schnellübung statt.

Gegeben seien die Klassen `Number` und `Rational` in den gleichnamigen Dateien auf der WWW-Seite zu dieser Übung.

1. Erweitern Sie die Klasse `Rational` um die Vergleichsoperatoren $<$, $<=$, $>$ und $>=$. Versuchen Sie dabei, für die Implementierung dieser Operatoren auf bereits zuvor implementierte Operatoren zurückzugreifen. Erweitern Sie die Datei `Main.cpp` um Beispielaufrufe für die obigen Operatoren. Testen Sie insbesondere die Funktionsweise für rationale Zahlen mit dem Wert 0 bzw. mit negativen Zählern und Nennern.

2. Führen Sie eine neue Klasse `Complex` für die Verarbeitung komplexer Zahlen ein, die wie `Rational` von der Oberklasse `Number` abgeleitet ist. Implementieren Sie den Konstruktor der Klasse, sowie die von der Basisklasse geforderten Ein- und Ausgaberoutinen. Fügen Sie die Operatoren $==$, $+$ und $*$ hinzu. Letztere sind für komplexe Zahlen $a_1 = (\alpha_1; \beta_1)$ und $a_2 = (\alpha_2; \beta_2)$ wie folgt definiert:

$$\begin{aligned} a_1 + a_2 &= (\alpha_1 + \alpha_2; \beta_1 + \beta_2) \\ a_1 \cdot a_2 &= (\alpha_1\alpha_2 - \beta_1\beta_2; \alpha_1\beta_2 + \alpha_2\beta_1) \end{aligned}$$

   Implementieren Sie zusätzlich eine Methode `isReal`, die überprüft, ob eine komplexe Zahl auch in der Menge der reellen Zahlen ist (d.h. ob der Imaginärteil gleich Null ist) und eine Methode `getReal`, die den Realteil zurückliefert. Erweitern Sie wiederum Ihre main-Funktion, um die Funktionsweise Ihrer neuen Klasse zu testen. Liefert die Methode `noOfNumbers` bei der Verwendung von `Rational`- und `Complex`-Objekten die richtige Antwort?

Die Dateien können unter Unix folgendermassen compiliert werden:

```
> g++ -c *.cpp
> g++ -o zahlen *.o
> ./zahlen
```

Abgabetermin: 27. Januar 2004