Hochschule Karlsruhe
University of
Applied Sciences

-SPO Zusatzaufgabe-

Informatik2

Rationale Brüche

Student:	Daniel Lipaj	75795
Universität:	Hochschule Karlsruhe	
Studiengang:	Elektro- und Informationstechnik	
Studienvertiefung:	Informationstechnik	
Semester:	Sommersemester 2022	
Dozent:	Prof. Dr. Thorsten Leize	

Inhaltsverzeichnis

1 Vorübelegungen

Es soll ein Programm in C++ geschrieben werden, mit der Klasse rational. So sollen rationale Brüche miteinander verrechnet, werden (Addiert, subtrahiert, multipliziert und dividiert). Als Vorüberlegung habe ich erster mal schrieftlich Brüche berechnet.

1.1 Addition

Fall 1.) Gleiche Nenner

$\frac{2}{9} + \frac{2}{9} = \frac{2+2}{9} = \frac{4}{9}$

Notiz: Es werden nur die Zähler addiert, ohne Erwiterung.

Fall 2.) Bruch1 Nenner ist ein Faktor von Bruch2 Nenner

$\frac{2}{3} + \frac{2}{9} = \frac{2}{3} \cdot \frac{3}{3} + \frac{2}{9} = \frac{6}{9} + \frac{2}{9} = \frac{6+2}{9} = \frac{8}{9}$

Notiz: Der Bruch1 wird mit einem Faktor erweitert, der entspricht

$\frac{NennerBruch2}{NennerBruch1} = \frac{9}{3} = 3$

Fall 3.) Bruch2 Nenner ist ein faktor von Bruch1 Nenner

Gleiches vorgehen wie im Fall 2.) Jedoch gilt:

$\frac{NennerBruch1}{NennerBruch2}$

Fall 4.) Die Nenner sind keine Faktoren voneinander

$\frac{2}{16} + \frac{2}{60} = \frac{2}{16} \cdot \frac{15}{15} + \frac{2}{60} \cdot \frac{4}{4} = \frac{30}{240} + \frac{8}{240} = \frac{30+8}{240} = \frac{38}{240}$

Notiz: Es muss der kleinste gemeinsame Vielfache bestimmt werden und mit einem faktor erweitert werden, der sich folgend bestimmen lässt:

$Faktor1 = \frac{kgV}{NennerBruch1} und\, Faktor2 = \frac{kgV}{Nennerruch2}$

1.2 Subtraktion

Analog mit der Addition.

1.3 Multipkikation

$\frac{2}{3} \cdot \frac{2}{3} = \frac{2 \cdot 2}{3 \cdot 3} = \frac{4}{9}$

Notiz: Es wird jeweils der Zähler und die Nenner miteinander multipliziert.

1.4 Division

$\frac{2}{3} \div \frac{2}{3} = \frac{2}{3} \cdot \frac{3}{2} = 1$

Notiz: Bruch1 wird mit dem kehrwert von Bruch2 multipliziert.

2 Quellcode Analyse

2.1 Klasse Rational

2.1.1 Privat/Public Section

```
1         calss rational
2         {
3             private:
4                 int zaehler, nenner;
5             public:
6                 ...
7         }
8
```

Die Klasse rational ist in zwei Bereichen aufgeteilt in privat und dem public Bereich. In dem private Bereich werden die Variablen Zähler und Nenner angelegt. Dadurch das die Variablen im Privatbereich abgekapselt sind, kann man nur innerhalb der Klasse auf diese zugreifen. Auf den public Bereich kann man auch außerhalb der Klasse rational zugreifen.

2.1.2 Konstruktor

```
1         class rational
2         {
3             privat:
4                 int zaehler, nenner;
5             public:
6                 rational(int zaehlerr, int nennerr)
7                 {
8                     setvar(zaehlerr, nennerr);
9                 }
10
11                void setvar(int zaehlerr, int nennerr)
12                {
13                    zaehler = zaehlerr;
14                    nenner = nennerr;
15                }
16
```

Der Konstruktor wird in der Klasse definiert, indem eine Funktion der Klasse genau so heißt wie die Klasse selber. Im Konstruktor wird die Funktion setvar innerhalb der Klasse aufgerufen. Die Funktion setvar soll kein Rückgabe Wert liefern, daher wird sie als void deklariert. Dadurch das die Funktion setvar innerhalb der Klasse definiert ist, darf sie auf die Variablen im privat Bereich zugreifen, druch diese Abkapselung ist ein ausversehen es setzen dieser Variablen o.ä ausgeschlossen.

2.1.3 Kleinster gemeinsames Vielfaches

```
1         int kgv(int nenner1, int nenner2)
2         {
3             int found_kgv = 0;
4             int i = 1;
5             while(found_kgv==0)
6             {
7                 if(i%nenner1==0)
8                 {
9                     if(i%nenner2==0)
10                    {
11                        found_kgv = 1;
12                    }
13                     else
14                     {
15                        i++;
16                     }
17                }
18                 else
19                 {
20                    i++;
21                 }
22            }
23            return i;
24        }
25
```

Das kleinste gemeinsame Vielfache wird benötiegt beim Erweitern der Brüche. Der Funktion werden die Parameter nenner1 und nenner2 jeweils als int Werte übergeben.
Es werden zwei int Variablen definiert found_kgv und i.
i als Lauf variable wird solange inkrementiert, bis die beiden Nenner der Brüche ganzzahlig durch i teilbar sind, wenn dies der Fall ist, ist der kleinste gemeinsame Vielfache bestimmt.

2.1.4 Größte gemeinsame Teiler

```
1         int ggt(int zaehler, int nenner)
2         {
3             if(nenner <0)
4             {
5                 nenner = nenner * (-1)
6             }
7
8             int i = nenner;
9             while(i <= nenner)
10            {
11                if(nenner%i==0)
12                {
13                    if(zaehler%i==0)
14                    {
15                        break;
16                    }
17                     else
18                     {
19                        i--;
20                     }
21                }
22                 else
23                 {
24                    i--;
25                 }
26            }
27            return i;
28        }
29
```

Der größte gemeinsame Teiler wird bestimmt zum Kürzen.
Der Funktion werden die Parameter zaehler und nenner als int Werte übergeben.
Falls der Übergebene Wert des Nenners Negativ sein sollte, wird dieser mit -1 multipliziert um ihn Positiv zu machen. Da wir i dekrementieren würde es ansonsten, ins undendliche gehen, an den Faktoren des Nenners ändert sich dadurch nichts.
i ist die Laufvariable, diese bekommt den anfangs Wert des Nennes und wird solange dekrementiert, bis i ganz zahlig durch den Zähler und Nenner teilbar ist.

2.1.5 Addition

```
1        rational operator +(rational bruch2)
2        {
3            if(nenner%bruch2.nenner == 0)
4            {
5                int multiplikator;
6
7                multiplikator = nenner/bruch2.nenner;
8                bruch2.zaehler = bruch2.zaehler * multiplikator;
9            }
10            else if(bruch2.nenner%nenner == 0)
11            {
12               int multiplikator;
13
14               multiplikator = bruch2.nenner/nenner;
15               zaehler = zaehler * multiplikator;
16            }
17             else
18             {
19               int kgv_;
20               int multiplikator;
21
22               kgv_ = kgv(nenner, bruch2.nenner);
23
24               multiplikator = kgv_/bruch2.nenner;
25               bruch2.zaehler = bruch2.zaehler * multiplikator;
26               bruch2.nenner = bruch2.nenner * multiplikator;
27             }
28             int ggt_;
29             ggt_ = ggt(zaehler+bruch2.zaehler, nenner);
30             if(ggt_ == 0)
31             {
32               return rational (zaehler+bruch2.zaehler, nenner);
33             }
34              else
35              {
36                   return rational ((zaehler+bruch2.zaehler)/ggt_,
    nenner/ggt_);
37              }
38
39       }
40
```

Im Fall 1. Ist der Nenner von Bruch2 ein Faktor von Bruch1, so muss nur der Bruch2 erweitert werden.
Im Fall 2. Ist der Nenner von Bruch1 ein Faktor von Bruch2, so muss nur der Bruch1 erweitert werden.
Bei allen anderen Fällen wird erster das kleinste gemeinsame vielfache berechnet, den wir brauchen, um den Faktor zu berechnen, mit dem jeweils die Brüche erweitert werden.
Nach dem erweitern wird der größte gemeinsame Teiler berechnet, falls dieser 0 sein sollte, wird der Bruch nicht gekürzt, ansonsten wird der Bruch gekürzt, indem wir Zähler und Nenner durch den ggT Dividieren.

2.1.6 Subtraktion

Analog zur Addition

2.1.7 Multiplikation

```
1         rational operator *(rational bruch2)
2         {
3             zaehler = zaehler * bruch2.zaehler;
4             nenner = nenner * bruch2.nenner;
5
6             int ggt_;
7             ggt_ = ggt(zaehler, nenner);
8             if(ggt_ == 0)
9             {
10                return rational(zaehler, nenner);
11            }
12             else
13             {
14                return rational(zaehler/ggt_, nenner/ggt_);
15             }
16        }
17
```

Für die Multiplikation werden die Zähler und die Nenner der Brüche jeweils miteinander multipliziert und anschließend der größte gemeinsame Teiler berechnet. Ist der ggT gleich null, wird der Bruch ohne Kürzen zurückgegeben. Wenn der ggt > 0 ist, wird der Zähler und der Nenner jeweils erster durch den ggT dividiert und anschließend zurückgegeben.

2.1.8 Division

```
1         rational operator /(rational bruch2)
2         {
3             zaehler = zaehler * bruch2.nenner;
4             nenner = nenner * bruch2.zaehler;
5
6             int ggt_ = ggt(zaehler, nenner);
7             if(ggt_ == 0)
8             {
9                 return rational(zaehler, nenner);
10            }
11             else
12             {
13                return rational(zaehler/ggt_, nenner/ggt_);
14             }
15        }
16
```

Der Bruch1 wird mit dem Kehrwert von Bruch2 multipliziert und anschließend wird wieder der ggT berechnet. Falls dieser null sein sollte, wird der Bruch ohne zu kürzen ausgegeben, ansonsten wird der Zähler und Nenner durch den ggT dividiert und ausgegeben.

2.1.9 Ausgabe

```
1         void print()
2         {
3             double kommazahl = double(zaehler)/double(nenner);
4
5             std::cout << zaehler << '/' << nenner << '\n';
6             std::cout << std::setprecision(3) << kommazahl << '\n';
7         }
8
```

Die Ausgabe Funktion wird als void definiert, da sie keinen Rückgabewert besitzt.
Anfangs wird die Dezimal Zahl des Bruches berechnet als double Wert. Dazu wird der Zähler und der Nenner zu double Werte gecastet und anschließend durcheinander dividiert.
Ausgegeben wird der Bruch und die dazugehörige Dezimalzahl.

2.2 Die Main-Funktion

2.2.1 Variablen Dekleration

```
1         int main()
2         {
3             int zaehler1, nenner1, zaehler2, nenner2;
4             char slash, operation;
5             rational bruch1(2, 3);
6             rational bruch2(2, 3);
7             rational bruch3(2, 3);
8             ...
9         }
10
```

Es werden die int Variablen zaehler1, nenner1, zaehler2 und nenner2 definiert, die jeweils die Eingabe Werte speichern werden.
Anschließend werden zwei char Variablen deklariert. Die Variable Slash wird im weiteren verlauf nur das Slash Zeichen in der Eingabe Abfangen und wird nicht weiter gebraucht. Die variable Operation nimmt das jeweilige zeichen der Operation auf und wird anschließend für eine Switch-Case Anweisung gebraucht.

2.2.2 Eingabe

```
1         int main()
2         {
3             ...
4             std::cout << "Eingabe: \n";
5             std::cin >> zaehler1 >> slash >> nenner1 >> operation
6             >> zaehler2 >> slash >> nenner2;
7             bruch1.setvar(zaehler1,nenner1);
8             bruch2.setvar(zaehler2,nenner2);
9             ...
10        }
11
```

Die Eingabe der Rechnung sieht folgend aus:
1/3 + 1/3 Für Positive Zahlen und -1/3 + 1/3 oder 1/3 + -1/3 für Negative Zahlen.
Zähler und Nenner werden in die jeweiligen variablen geschrieben, so wie zweimal der Slash in die Slash variable und der Operator und in die Operation Variable.

2.2.3 Switch-Case

```
1   int main()
2   {
3       ...
4       switch(operation)
5       {
6           case '+': bruch3 = bruch1 + bruch2;
7                     bruch3.print();
8                     break;
9           case '-': bruch3 = bruch1 - bruch2;
10                    bruch3.print();
11                    break;
12          case '*': bruch3 = bruch1 * bruch2;
13                    bruch3.print();
14                    break;
15          case '/': bruch3 = bruch1 / bruch2;
16                    bruch3.print();
17                    break;
18      }
19      return 0;
20  }
21
```

Der Switch-Case Anweisung wird die variable operation gegeben, mit der anschließend "gefiltert"wird. Dadurch das die vier operationszeichen überladen sind, werden bei der Berechnung der zwei Klassen die jeweilige Funktion aufgerufen und der return wird in Bruch3 gespeichert. Am Ende wird die Funktion print aus der rationalen Klasse ausgerufen, um das Ergebnis auszugeben

2.3 Tests

Es wurden verschiedene Aufgaben mit rationalen Brüche durchgeführt und das Ergebnis sollte ein Bruch sein, der weitestgehend gekürzt ist.

2.3.1 Addition

1.) $\frac{2}{6} + \frac{2}{7} = \frac{13}{21}\surd$ 2.) $\frac{2}{8} + \frac{4}{9} = \frac{25}{36}\surd$ 3.) $\frac{2}{8} + \frac{2}{3} = \frac{11}{12}\surd$

4.) $\frac{2}{5} + \frac{2}{7} = \frac{24}{35}\surd$ 5.) $\frac{4}{5} + \frac{-1}{3} = \frac{7}{15}\surd$ 6.) $\frac{-3}{2} + \frac{3}{5} = \frac{-9}{10}\surd$

7.) $\frac{-4}{9} + \frac{1}{6} = \frac{-5}{18}\surd$ 8.) $\frac{-2}{3} + \frac{-1}{2} = \frac{-7}{6}\surd$ 9.) $\frac{-4}{15} + \frac{-5}{6} = \frac{-11}{10}\surd$

2.3.2 Subtraktion

1.) $\frac{2}{4} - \frac{3}{9} = \frac{1}{6}\surd$ 2.) $\frac{5}{6} - \frac{3}{5} = \frac{7}{30}\surd$ 3.) $\frac{8}{9} - \frac{4}{7} = \frac{20}{63}\surd$

4.) $\frac{2}{3} - \frac{4}{8} = \frac{1}{6}\surd$ 5.) $\frac{2}{3} - \frac{-3}{5} = \frac{19}{15}\surd$ 6.) $\frac{7}{12} - \frac{-1}{20} = \frac{19}{30}\surd$

7.) $\frac{-5}{16} - \frac{-5}{18} = \frac{1}{18}\surd$ 8.) $\frac{5}{6} - \frac{2}{3} = \frac{1}{6}\surd$ 9.) $\frac{-3}{22} - \frac{2}{33} = \frac{-13}{66}\surd$

2.3.3 Multiplikation

1.) $\frac{6}{12} \cdot \frac{5}{15} = \frac{1}{6}\surd$ 2.) $\frac{12}{15} \cdot \frac{3}{6} = \frac{2}{5}\surd$ 3.) $\frac{4}{21} \cdot \frac{7}{16} = \frac{1}{12}\surd$

4.) $\frac{12}{14} \cdot \frac{14}{18} = \frac{2}{3}\surd$ 5.) $\frac{14}{24} \cdot \frac{12}{28} = \frac{1}{4}\surd$ 6.) $\frac{8}{14} \cdot \frac{7}{24} = \frac{1}{6}\surd$

7.) $\frac{4}{20} \cdot \frac{12}{20} = \frac{3}{25}\surd$ 8.) $\frac{7}{11} \cdot \frac{4}{9} = \frac{28}{99}\surd$ 9.) $\frac{-5}{9} \cdot \frac{-9}{10} = \frac{1}{2}\surd$

2.3.4 Division

1.) $\frac{10}{50} \div \frac{5}{10} = \frac{2}{5}\surd$ 2.) $\frac{3}{9} \div \frac{6}{14} = \frac{7}{9}\surd$ 3.) $\frac{18}{49} \div \frac{6}{21} = \frac{9}{7}\surd$

4.) $\frac{20}{70} \div \frac{30}{70} = \frac{2}{3}\surd$ 5.) $\frac{4}{16} \div \frac{7}{21} = \frac{3}{4}\surd$ 6.) $\frac{8}{25} \div \frac{8}{35} = \frac{7}{5}\surd$

7.) $\frac{-3}{8} \div \frac{9}{2} = \frac{-1}{12}\surd$ 8.) $\frac{5}{8} \div \frac{-3}{4} = \frac{-5}{6}\surd$ 9.) $\frac{-5}{12} \div \frac{1}{6} = \frac{-5}{2}\surd$

3 Fazit

Die Vorlesung Informatik2 habe ich bei Frau Katz besucht, leider ist C++ nur relativ wenig Bestandteil der Vorlesung bei ihr. Daher hab ich mit C++ bisher nur relativ wenig Berührungspunkte gehabt. Neben C schreibe ich seit Längerem Python, indem man auch Objektorientiertes programmieren kann, jedoch hab ich dies nur sehr wenig benutzt.
Allerdings hat mich dieses kleine Projekt, C++ näher gebracht und dadurch werde ich mich jetzt auch weiterhin etwas mehr mit C++ beschäftiegen.