Wissenschaftliches Rechnen mit C++

Aufgabeblatt 2

LOICH KAMDOUM DEAMENI
Mat-Nr. 506520

1. Die geeignete Struktur für rationale Nummer ist eine Klasse mit benötigten Methoden zur Benutzung der Nummer.
private:
	int zaehler;
	int nenner;

2.
Rational.h

```
/*
 * Rational.h
 *  Created on: 19.04.2019
 *      Author: kamdoum
 */

#include <iostream>

#ifndef RATIONAL_H_
#define RATIONAL_H_

class Rational {

public:
	//Konstruktoren
	Rational();
	Rational(int);
	Rational(int, int);

	//Destruktor
	virtual~ Rational();

	//Methoden
	int numerator();
	int denominator();
	int gcd(int, int);
	double toDouble();
	void print();

	//Operatoren
	Rational &operator*=(const Rational &);
	Rational &operator+=(const Rational &);
	Rational &operator-=(const Rational &);
	Rational &operator/=(const Rational &);
```

```
	//Vergleichsoperatoren
	friend bool operator==(const Rational &, const Rational &);
	friend bool operator!=(const Rational &, const Rational &);
	friend bool operator<=(const Rational &, const Rational &);
	friend bool operator>=(const Rational &, const Rational &);
	friend bool operator<(const Rational &, const Rational &);
	friend bool operator>(const Rational &, const Rational &);

	//Operatoren
	friend Rational operator*(const Rational &, const Rational &);
	friend Rational operator+(const Rational &, const Rational &);
	friend Rational operator-(const Rational &, const Rational &);
	friend Rational operator/(const Rational &, const Rational &);

private:
	int zaehler;
	int nenner;

};

#endif /* RATIONAL_H_ */
```

Rational.cpp

```
//============================================================================
// Name        : Rational.cpp
// Author      : loich kamdoum deameni
// Version     :
// Copyright   : Your copyright notice
// Description : Programm in C++, Ansi-style
//============================================================================
#include "Rational.h"

using namespace std;

Rational::Rational(){
	zaehler = 1;
	nenner = 1;
}

Rational::Rational(int zaehler_){
	zaehler = zaehler_;
	nenner = 1;
}

Rational::Rational(int zaehler_, int nenner_){
	zaehler = zaehler_;
	nenner = nenner_;
}

Rational::~Rational(){
```

```
}

int Rational::numerator(){
	return zaehler;
}

int Rational::denominator(){
	return nenner;
}

void Rational::print(){
	cout << zaehler<< " / "<< nenner;
}

int Rational::gcd(int zaehler_, int nenner_){
	if (nenner_ == 0)
	  return zaehler_;
	return gcd(nenner_, zaehler_ % nenner_);
}

// Ergibt die numerator in double.
double Rational::toDouble(){
	return (double) this->numerator();
}

Rational &Rational::operator*=(const Rational &rational_){
	zaehler *= rational_.zaehler;
	nenner *= rational_.nenner;
	int gcd_ =  gcd(zaehler, nenner);
	*this = Rational(zaehler/gcd_, nenner/gcd_);
	return *this;
}

Rational &Rational::operator/=(const Rational &rational_){
	zaehler *= rational_.nenner;
	nenner *= rational_.zaehler;
	int gcd_ =  gcd(zaehler, nenner);
	*this = Rational(zaehler/gcd_, nenner/gcd_);
	return *this;
}

Rational &Rational::operator+=(const Rational &rational_){
	zaehler = this->numerator()*rational_.nenner + rational_.zaehler*this->denominator();
	nenner = this->denominator()*rational_.nenner;
	int gcd_ =  gcd(zaehler, nenner);
	*this = Rational(zaehler/gcd_, nenner/gcd_);
	return *this;
}

Rational &Rational::operator-=(const Rational &rational_){
	zaehler = this->numerator()*rational_.nenner - rational_.zaehler*this->denominator();
	nenner = this->denominator()*rational_.nenner;
```

```
	int gcd_ =  gcd(zaehler, nenner);
	*this = Rational(zaehler/gcd_, nenner/gcd_);
	return *this;
}

Rational operator+(const Rational &rational_links, const Rational &rational_rechts){
	Rational tmp = rational_links;
	return tmp+=rational_rechts;
}

Rational operator-(const Rational &rational_links, const Rational &rational_rechts){
	Rational tmp = rational_links;
	return tmp-=rational_rechts;
}

Rational operator*(const Rational &rational_links, const Rational &rational_rechts){
	Rational tmp = rational_links;
	return tmp*=rational_rechts;
}

Rational operator/(const Rational &rational_links, const Rational &rational_rechts){
	Rational tmp = rational_links;
	return tmp/=rational_rechts;
}

bool operator==(const Rational &rational_links, const Rational &rational_rechts){
	Rational tmp = Rational();
	int gcd_links =  tmp.gcd(rational_links.zaehler, rational_links.nenner);
	int gcd_rechts =  tmp.gcd(rational_rechts.zaehler, rational_rechts.nenner);
	Rational links = Rational(rational_links.zaehler/gcd_links, rational_links.nenner/gcd_links);
	Rational rechts = Rational(rational_rechts.zaehler/gcd_rechts,
rational_rechts.nenner/gcd_rechts);
	return ((links.zaehler == rechts.zaehler)
			&& (links.nenner == rechts.nenner));
}

bool operator!=(const Rational &rational_links, const Rational &rational_rechts){
	return !(rational_links == rational_rechts);
}

bool operator<=(const Rational &rational_links, const Rational &rational_rechts){
	return ((rational_links < rational_rechts) || (rational_links == rational_rechts));
}

bool operator>=(const Rational &rational_links, const Rational &rational_rechts){
	return ((rational_links > rational_rechts) || (rational_links == rational_rechts));
}

bool operator<(const Rational &rational_links, const Rational &rational_rechts){
	int links =  rational_links.zaehler*rational_rechts.nenner;
	int rechts =  rational_links.nenner*rational_rechts.zaehler;
	return (links < rechts);
```

```
}

bool operator>(const Rational &rational_links, const Rational &rational_rechts){
	int links =  rational_links.zaehler*rational_rechts.nenner;
	int rechts =  rational_links.nenner*rational_rechts.zaehler;
	return (links > rechts);
}
```

Ueberprufung.cpp

```
/*
 * Ueberprufung.cpp
 *  Created on: 19.04.2019
 *      Author: kamdoum
 */

#include <iostream>
#include "Rational.h"

using namespace std;

int main ( int argc, char *argv[] );

int main ( int argc, char *argv[] ){

	//Initialisierung
	int gcd_ = 0;
	Rational f1 = Rational(-3, 12);
	Rational f2 = Rational(4, 3);
	Rational f3 = Rational(0, 1);

	/*********************************************/
	//Testen
	cout <<'\n'<<"Testen"<<'\n';
	f3 = f1+f2;
	cout << "f1+f2 = ";
	f3.print();
	cout << '\n';

	f3 = f1*f2;
	cout << "f1*f2 = ";
	f3.print();
	cout << '\n';

	f3 = 4+f2;
	cout << "4+f2 = ";
	f3.print();
	cout << '\n';

	f3 = f2+5;
	cout << "f2+5 = ";
	f3.print();
```

```
	cout << '\n';

	f3 = 12*f1;
	cout << "12*f1 = ";
	f3.print();
	cout << '\n';

	f3 = f1*6;
	cout << "f1*6 = ";
	f3.print();
	cout << '\n';

	f3 = f1/f2;
	cout << "f1/f2 = ";
	f3.print();
	cout << '\n';

	/**************************************************/
	//Test von Vergleichsoperatoren
	cout <<'\n'<<"Test von Vergleichsoperatoren"<<'\n';
	f1 = Rational(3, 12);
	f2 = Rational(1, 4);

	f1.print();
	cout << " (";

	if(f1 == f2)
		cout <<" == ";
	if(f1 != f2)
		cout <<" != ";
	if(f1 <= f2)
		cout <<" <= ";
	if(f1 >= f2)
		cout <<" >= ";
	if(f1 < f2)
		cout <<" < ";
	if(f1 > f2)
		cout <<" > ";

	cout << ") ";
	f2.print();
	cout << '\n';

	/**************************************************/
	cout <<'\n'<<"Test von toDouble() mit f1 "<<'\n';
	cout << "f1.toDouble() = "<< f1.toDouble();
	cout << '\n';

	cout << '\n'<<"Ende des Programms" << '\n';

}
```