

10.12.2022

SWE3 Übung 04


s2010458016

MARCO SARKADY

WS 22/23

GESAMTAUFWAND: 10 STD

Medizin – und Bioinformatik



Inhalt

Lösungsidee	3
Testfälle	4

Lösungsidee

Aufgabe ist es, die letzte Übung in einen generische Klasse umzuwandeln. Die Ausimplementierung der Funktionen findet im Headerfile statt (wegen Template). Außerdem funktionieren die Operatorenüberladungen mit dem Barton Nack-Man Trick (Stichwort friend inline).

Außerdem ändert sich die Struktur auch, wie im Vergleich zur letzten Übung. Für die Operatorüberladungen benötigt man außer der rhs, die lhs, um auf die „Linke Seite“ zugreifen zu können.

Weil es einige Fälle gibt, wo man eine Zahl auf Null oder Eins prüft, wird ein Operations File erstellt, und Funktionen implementiert, die dieses überprüfen. Die Funktion Zero retourniert z.B. eine 1, weil das Neutrale Element für Rationale Zahlen die eins ist.

Zudem wird ein Number file erstellt und ausschließlich mit dem Objekt Rational<Number<int>> getestet.

Mit einem selbst definiertem Datentyp funktioniert die Methode to_string nicht mehr, ich habe mich entschieden diese Funktion so zu lassen wie sie war.

Die Funktion abs soll eigentlich überprüfen, ob die übergebene Zahl negativ ist. Wenn ja, wird die Zahl * -1 multipliziert und retourniert, andernfalls wird die gleiche Zahl retourniert.

In der Funktion divides wird überprüft, ob 'a % b' das neutrale Element null des Typen T ergibt. Wenn ja, bedeutet das, dass a b teilt und daher wird true retourniert.

In der Funktion equals wird überprüft, ob die beiden Parameter gleich sind.

Die Funktion gcd macht dasselbe, wie die normalize_rec Funktion. Hier wird also der greatest common denominator berechnet, in dem man eine rekursive Funktion baut. Der gcd wird dann retourniert.

Die Funktion negate retourniert die übergebene Zahl, in dem ein – vorne darangehängt wird.

Die Funktion remainder retourniert das Ergebnis der Modularechnung von a und b, da dies den Rest berechnet.

Testfälle

Test Comparison Operators

```
test equal to
2:1
2:1
true

test unequal
2:1
2:1
false

test smaller than
2:1
2:1
false

test smaller than or equal to
2:1
2:1
true

test greater than
2:1
2:1
false

test greater than or equal to
2:1
2:1
true
```

```
test equal to
8:5
2:1
false

test unequal
8:5
2:1
true

test smaller than
8:5
2:1
true

test smaller than or equal to
8:5
2:1
true

test greater than
8:5
2:1
false

test greater than or equal to
8:5
2:1
false
```

Arithmetic Operators

Test addition

2:1

5:2

result: 9:2

Test subtraction

2:1

5:2

result: 1:-2

Test multiplication

2:1

5:2

result: 5:1

Test division

2:1

5:2

result: 4:5

test division by zero

Divide By Zero Error

Compound Assignment Operators

Test addition (compound assignment operator)

2:1

5:2

result: 9:2

Test subtraction (compound assignment operator)

2:1

5:2

result: 1:-2

Test multiplication (compound assignment operator)

2:1

5:2

result: 5:1

Test division (compound assignment operator)

2:1

5:2

result: 4:5

test division by zero (compound assignment operator)

Divide By Zero Error

Test other functions

```
test is_pos, neg, zero
2:1
is positive: true
is negative: false
is zero: false
0:1
is positive: true
is negative: false
is zero: true
-2:1
is positive: false
is negative: true
is zero: false
```

```
test getter
2:1
numerator: 2
denominator: 1
```

```
test inverse
2:1
1:2
```

```
test copy constructor
0:1
5:1
a = b
5:1
```

```
test scan function
Enter two numbers:
4
2
4:2

*****
```