МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

"САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО"

Институт компьютерных наук и технологий

Направление **02.03.01** : Математика и компьютерные науки

ОТЧЕТ О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ 4

Исполнитель: _____ Яшнова Дарья Михайловна
группа 5130201/20002

Руководитель: _____ Моторин Дмитрий Евгеньевич

«___»_____ 2024г

Санкт-Петербург, 2024

Введение

В данном практическом нужно разработать библиотеку функций на языке программирования Haskell с использованием фреймворка для модульного тестирования QuickCheck. Библиотека должна содержать две функции: multiplyMod и gcd.

Функция multiplyMod должна вычислять произведение двух чисел по модулю заданного числа. Реализация функции multiplyMod должна удовлетворять нескольким важным свойствам, таким как нейтральный элемент и коммутативность.

Функция gcd должна вычислять наибольший общий делитель двух целых чисел. Реализация функции gcd должна удовлетворять таким свойствам, как равенство числа самому себе, коммутативность и ассоциативность.

Содержание

1 Описание математических функций

1.1 Функция mGcd

Для вычисления наибольшего общего делителя двух чисел реализована функция mGcd.

```
   mGcd :: Int -> Int -> Int
mGcd a b
 | a == 0 = abs b
 | b == 0 = abs a
 | a == b = b
 | otherwise = mGcd' (abs a) (abs b)
 where
  mGcd' a 0 = a
  mGcd' a b = mGcd' b (a 'mod' (b))
```

Тип: mGcd принимает два целых числа (Int) и возвращает одно целое число (Int), которое является их наибольшим общим делителем (НОД).

Если одно из чисел равно нулю, функция возвращает абсолютное значение другого числа.

Если оба числа равны, функция возвращает одно из них.

В противном случае вызывается вспомогательная рекурсивная функция mGcd', которая принимает абсолютные значения входных чисел.

Вспомогательная функция mGcd':

Если второе число равно нулю, она возвращает первое число как НОД.

В противном случае она рекурсивно вызывает себя, передавая второе число и результат выражения (a mod (b)).

1.2 Функция modMult

```
modMult :: Int -> Int -> Int -> Int
modMult a b c = mod (a * b) c
```

Тип: modMult принимает три целых числа (Int) и возвращает одно целое число (Int).

Эта функция вычисляет произведение первого числа a на сумму второго числа b и единицы, а затем находит остаток от деления этого произведения на третье число c.

Используется встроенная функция mod, которая возвращает остаток от деления.

2 Описание тестирующего модуля

2.1 Главная функция main

```
main :: IO ()
main = do
 putStrLn "GCD tests"
 quickCheckWith stdArgs { maxSuccess = 1000 }  gcd_selfcheck
 putStrLn "1!"
 quickCheckWith stdArgs { maxSuccess = 1000 }  gcd_one_left_check
 putStrLn "2!"
 quickCheckWith stdArgs { maxSuccess = 1000 }  gcd_one_right_check
 putStrLn "3!"
```

```
quickCheckWith stdArgs { maxSuccess = 1000 }  gcd_comm_check
putStrLn "4!"
quickCheckWith stdArgs { maxSuccess = 1000 }  gcd_associative_check
putStrLn "Готово!"
putStrLn "mul tests"
quickCheckWith (stdArgs {maxSuccess = 1000}) $
   forAll (choose (-100, 100)) (\x ->
    forAll (choose (-100, 100)) (\y ->
     forAll nonZero (\m -> mul_mod_check x y m)))
quickCheckWith (stdArgs {maxSuccess = 1000}) $
  forAll (choose (-100, 100)) (\x ->
   forAll nonZero (\m -> mul_neutral_r_check x m))
quickCheckWith (stdArgs {maxSuccess = 1000}) $
  forAll (choose (-100, 100)) (\x ->
   forAll (choose (-100, 100)) (\y ->
    forAll nonZero (\m -> mul_comm_check x y m)))
putStrLn "Готово!"
```

В функции main выводятся сообщения о начале тестов, после чего выполняются различные проверки свойств функций mGcd и modMult. Используется quickCheckWith, чтобы задать параметры тестирования, такие как максимальное количество успешных попыток (maxSuccess = 1000).

2.2 Тесты для функции mGcd

- gcd_selfcheck

```
gcd_selfcheck :: Int -> Bool
gcd_selfcheck a = mGcd a a == a
```

Проверяет, что НОД числа с самим собой равен этому числу.

- gcd_one_left_check и gcd_one_right_check

```
gcd_one_left_check :: Int -> Bool
gcd_one_left_check a = mGcd 1 a == 1

gcd_one_right_check :: Int -> Bool
gcd_one_right_check a = mGcd a 1 == 1
```

Проверяет, что НОД единицы и любого числа всегда равен единице.

- gcd_comm_check

```
 gcd_comm_check :: Int -> Int -> Bool
gcd_comm_check a b = (mGcd a b) == (mGcd b a)
```

Проверяет коммутативность НОД: порядок чисел не влияет на результат.

- gcd_associative_check

```
gcd_associative_check :: Int -> Int -> Int -> Bool
gcd_associative_check a b c = mGcd (mGcd a b) c == mGcd a (mGcd b c)
```

Проверяет ассоциативность НОД: порядок вычисления не влияет на результат.

2.3 Тесты для функции modMult

- mul_mod_check

```
 mul_mod_check :: Int -> Int -> Int -> Bool
mul_mod_check x y m = mod (modMult x y m) m == mod (x*y) m
```

Проверяет, что результат произведения по модулю совпадает с произведением двух чисел по модулю.

- mul_neutral_r_check

```
mul_neutral_r_check :: Int -> Int -> Bool
mul_neutral_r_check x m = mod x m == modMult x 1 m
```

Проверяет, что умножение на единицу не изменяет число.

- mul_comm_check

```
mul_comm_check :: Int -> Int -> Int -> Bool
mul_comm_check x y m = modMult y x m == modMult x y m
```

Проверяет коммутативность операции умножения по модулю.

2.4 Генераторы и тестовые данные

```
nonZero :: Gen Int
nonZero = choose (-100, 100) suchThat (/= 0)
```

nonZero — это генератор случайных целых чисел в диапазоне от -100 до 100, исключая ноль. Он используется в тестах, где необходимо гарантировать, что одно из чисел не равно нулю.

3 Результаты

На рис.1 представлены результаты тестирования Lib.hs.

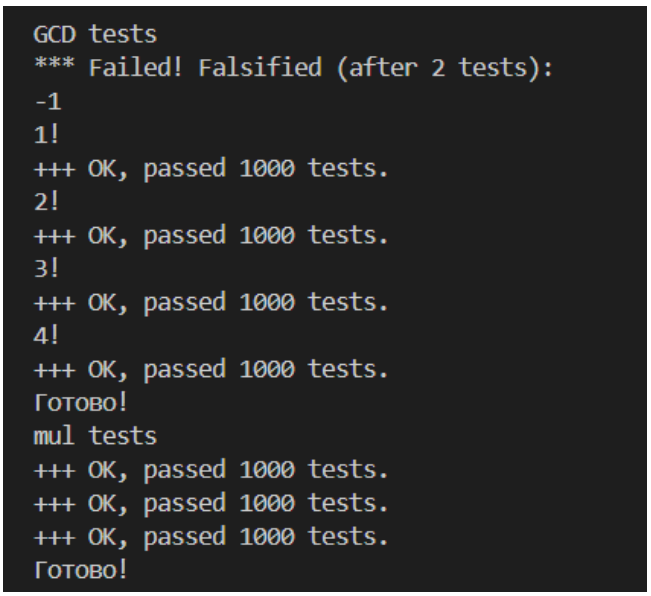

Рис. 2: Результат тестирования функции mGcd с ошибкой

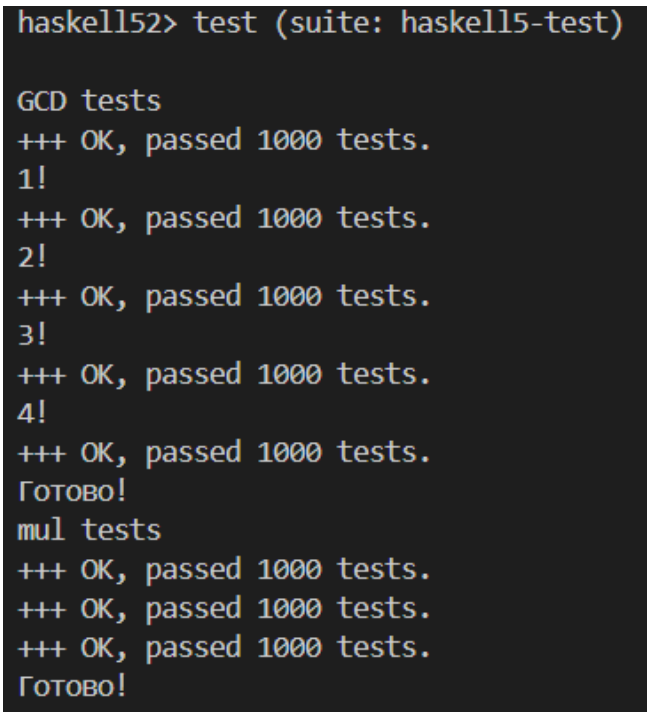

Рис. 1: Результаты тестирования модуля Lib.hs

Функцию mGcd можно изменить следующим образом:

```
mGcd :: Int -> Int -> Int
mGcd 0 0 = 0  -- Неправильный результат
mGcd a b = gcd a b  -- Правильная реализация для остальных случаев
```

Результат тестирования представлен на рис.2.

Функция не прошла тест, проверяющий что НОД числа с самим собой равен этому числу.

Выводы

В результате выполнения работы была изучена библиотека QuickCheck. Так же были разработаны тесты для функций multiplyMod и gcd. Были созданы тесты, проверяющие следующие свойства данных функций:

- нейтральный элемент для функции multiplyMod;
- коммутативность для функций multiplyMod и gcd;
- ассоциативность для функции gcd. Тесты были запущены с использованием команды stack test. Были успешно выполнены все написанные тесты, что подтверждает корректность реализации функций multiplyMod и gcd.

Список литературы

1. W. Kurt. Get Programming with Haskell. Москва: ДМК, 2019.
   URL: https://www.gutenberg.org/files/67882/67882-h/67882-h.htm