|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 5 |

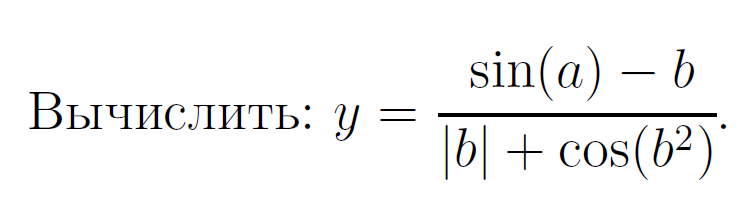
**Дисциплина:** Языки интернет-программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-35 Б |  | Scan0026 | И.А. Дулина |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Е.Ю. Гаврилова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Вариант 8**

**Часть 1**

**Задание:** 

**Код программы:**

*Файл пользователя (client.rb)*:

*# frozen\_string\_literal: true*require './main'  
puts('Введите a')  
*a* = gets.chomp  
puts('Введите b')  
*b* = gets.chomp  
puts('Y: ')  
puts(calc(*a*, *b*))

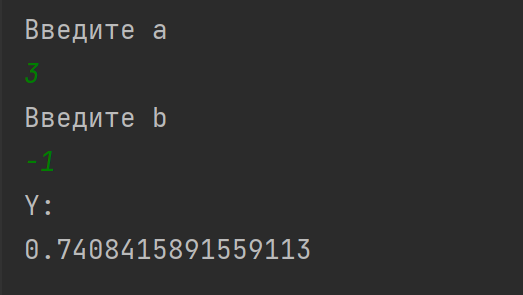
*Файл основной программы (main.rb):*

*# frozen\_string\_literal: true*def calc(*aaa*, *bbb*)  
 (Math.sin(*aaa*.to\_f) - *bbb*.to\_f) / (*bbb*.to\_f.abs + Math.cos(*bbb*.to\_f \* *bbb*.to\_f))  
end

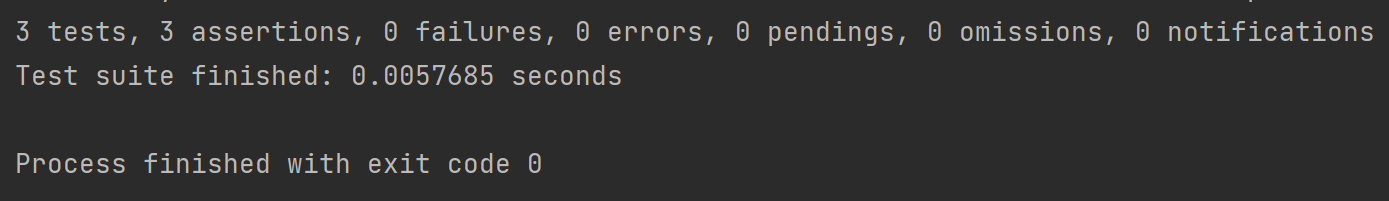
*Файл тестов (test.rb):*

*# frozen\_string\_literal: true*require 'minitest/autorun'  
require './main'  
  
class *TestCalc* < MiniTest::Test  
 def test\_calc  
 assert\_in\_delta(-0.925, calc(10, 5), 0.01)  
 assert\_in\_delta(-1.225, calc(-3.5, 2), 0.01)  
 assert\_in\_delta(0.574, calc(11, -1.5), 0.01)  
 end  
end

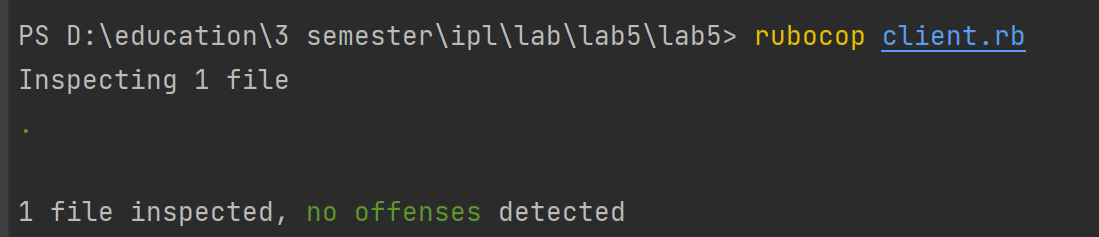
**Результат работы программы:**

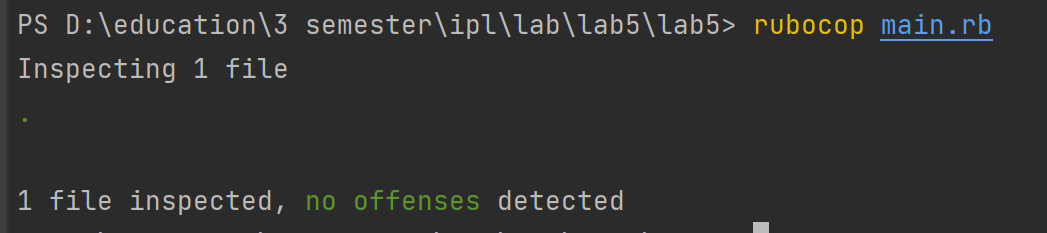


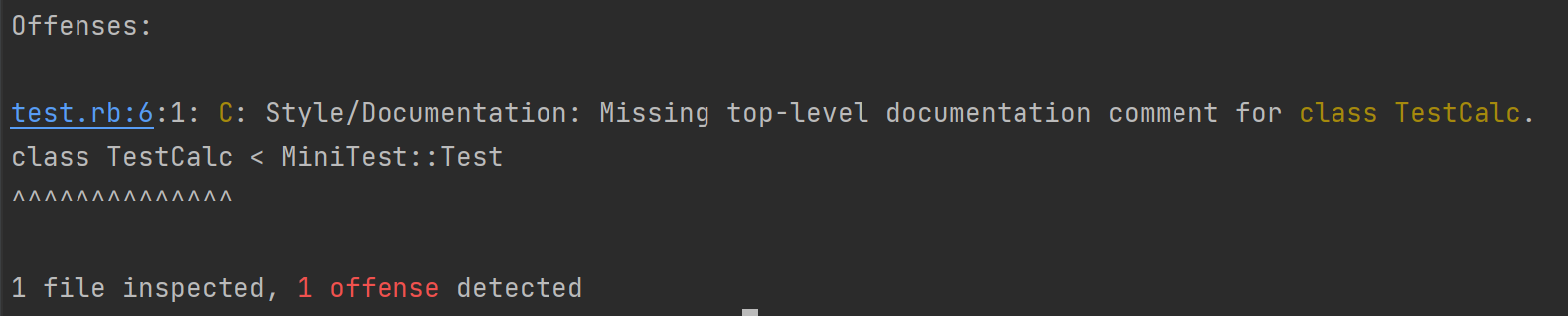
**Результат работы тестов**:



**Результат работы rubocop:**

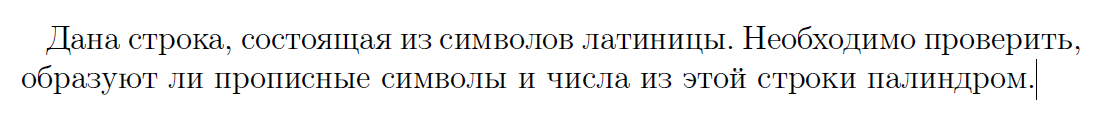






**Часть 2**

**Задание:**



**Код программы:**

*Файл пользователя (client2.rb):*

*# frozen\_string\_literal: true*require './main2'  
puts('Введите строку')  
*s* = gets.chomp  
if prov(*s*) == true  
 puts('Является палиндромом')  
else  
 puts('Не является палиндромом')  
end

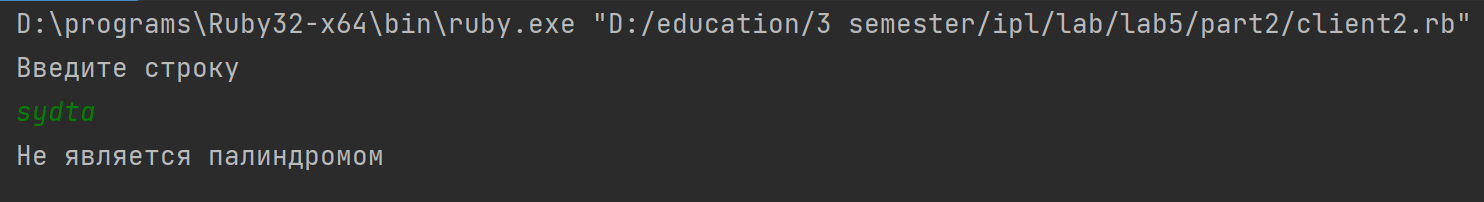
*Файл основной программы (main2.rb):*

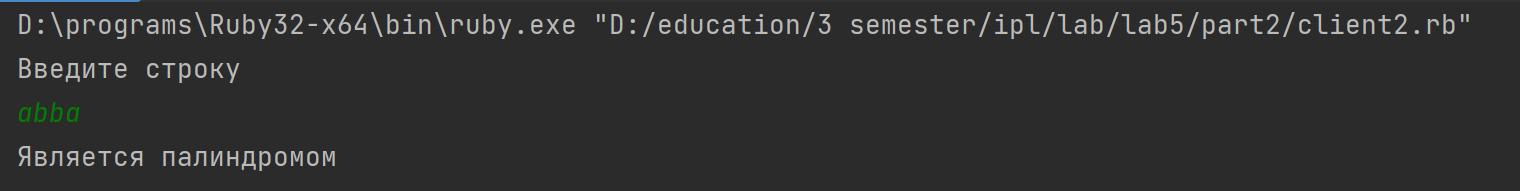
*# frozen\_string\_literal: true*def prov(*str*)  
 *str* == *str*.reverse  
end

*Файл тестов (test2.rb):*

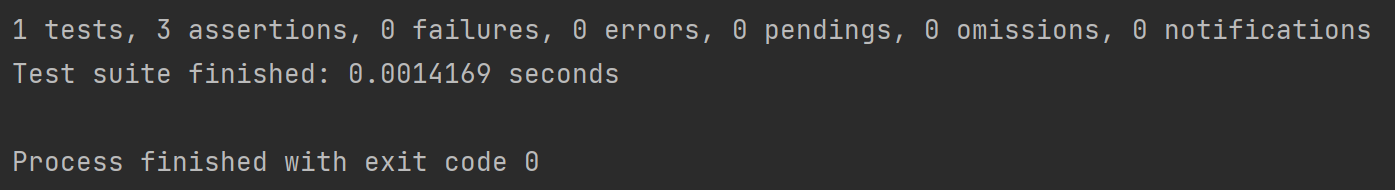
*# frozen\_string\_literal: true  
  
require 'test/unit'  
require './main2'  
  
class TestPal < Test::Unit::TestCase  
 def test\_pal  
 assert\_equal(true, prov('gretypyterg'))  
 assert\_equal(false, prov('idfdsfbk'))  
 assert\_equal(true, prov('typpyt'))  
 end  
end*

**Результат работы программы:**

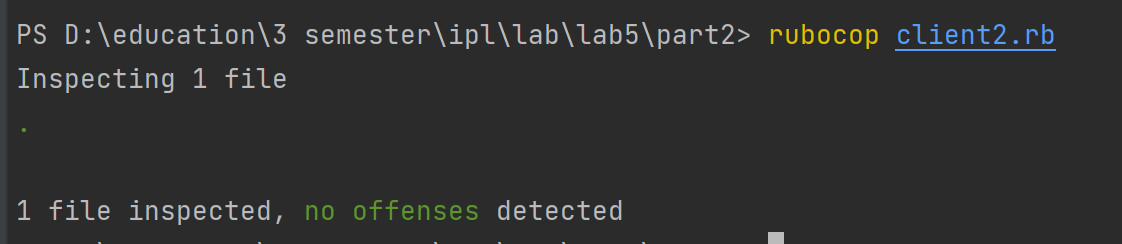


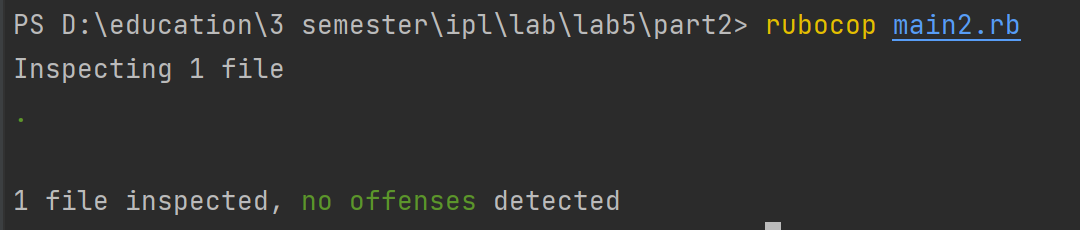


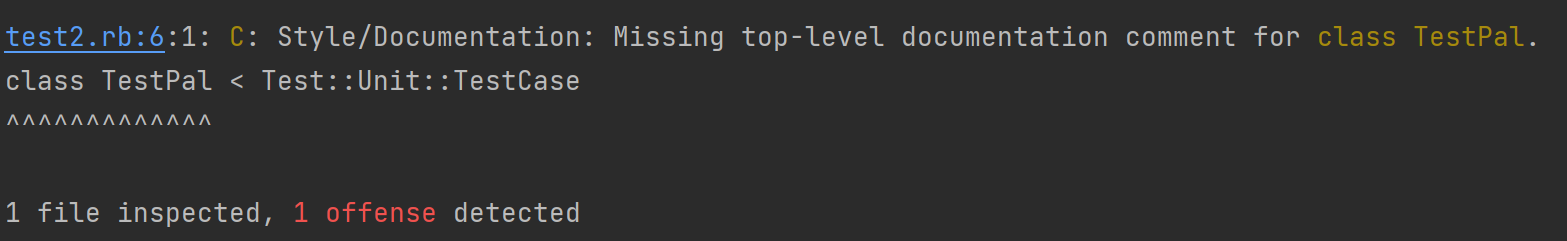
**Результат работы тестов:**



**Результат работы rubocop:**

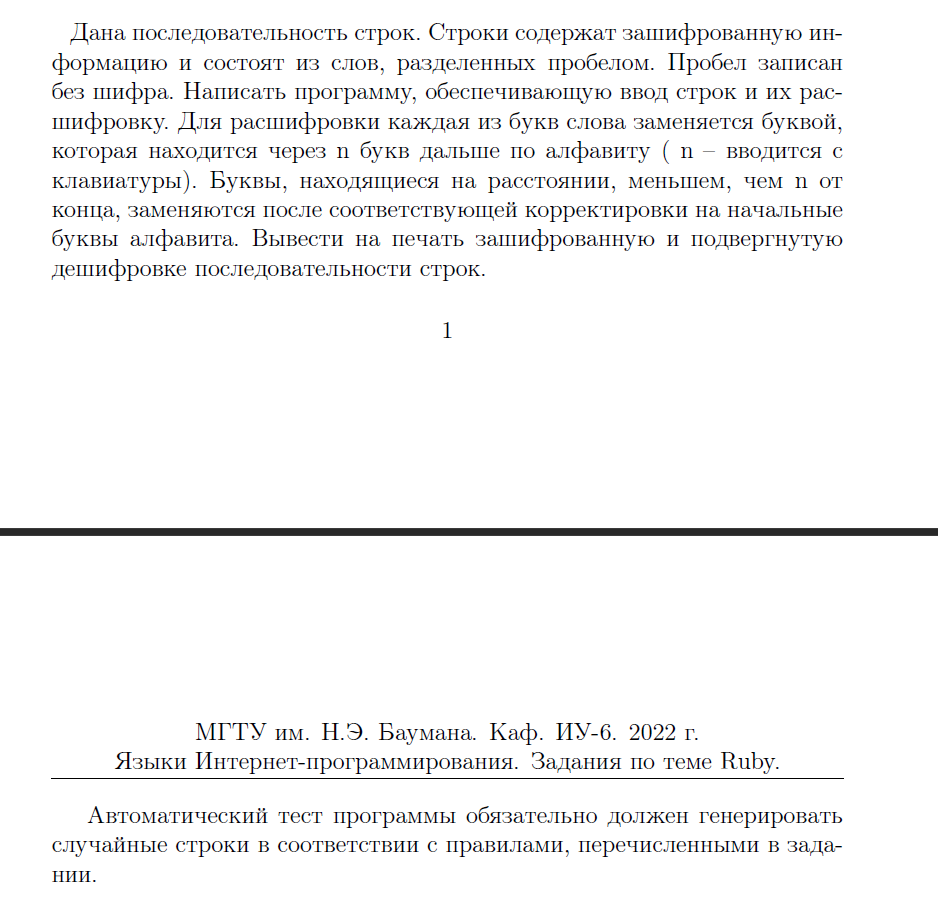






**Часть 3**

**Задание:**



**Код программы:**

Файл пользователя (client.rb):

*# frozen\_string\_literal: true  
  
require './main'  
puts('Введите n - на сколько букв вперёд будет производиться замена')  
n = gets.chomp.to\_i  
puts('Введите количеcтво строк')  
count = gets.chomp.to\_i  
puts('Введите каждую из них, разделяя слова пробелами')  
massive = []  
massive\_former=[]  
i = 0  
count.times do*

*str = gets.chomp*

*str.tr!('0-9', '')*

*str.downcase!*

*massive\_former << str*

*massive << shifr(str, n)*

*end  
puts('Зашифрованные строки: ')  
massive\_former.each { |s| puts s }  
puts('Раcшифрованные строки: ')  
massive.each { |s| puts s }*

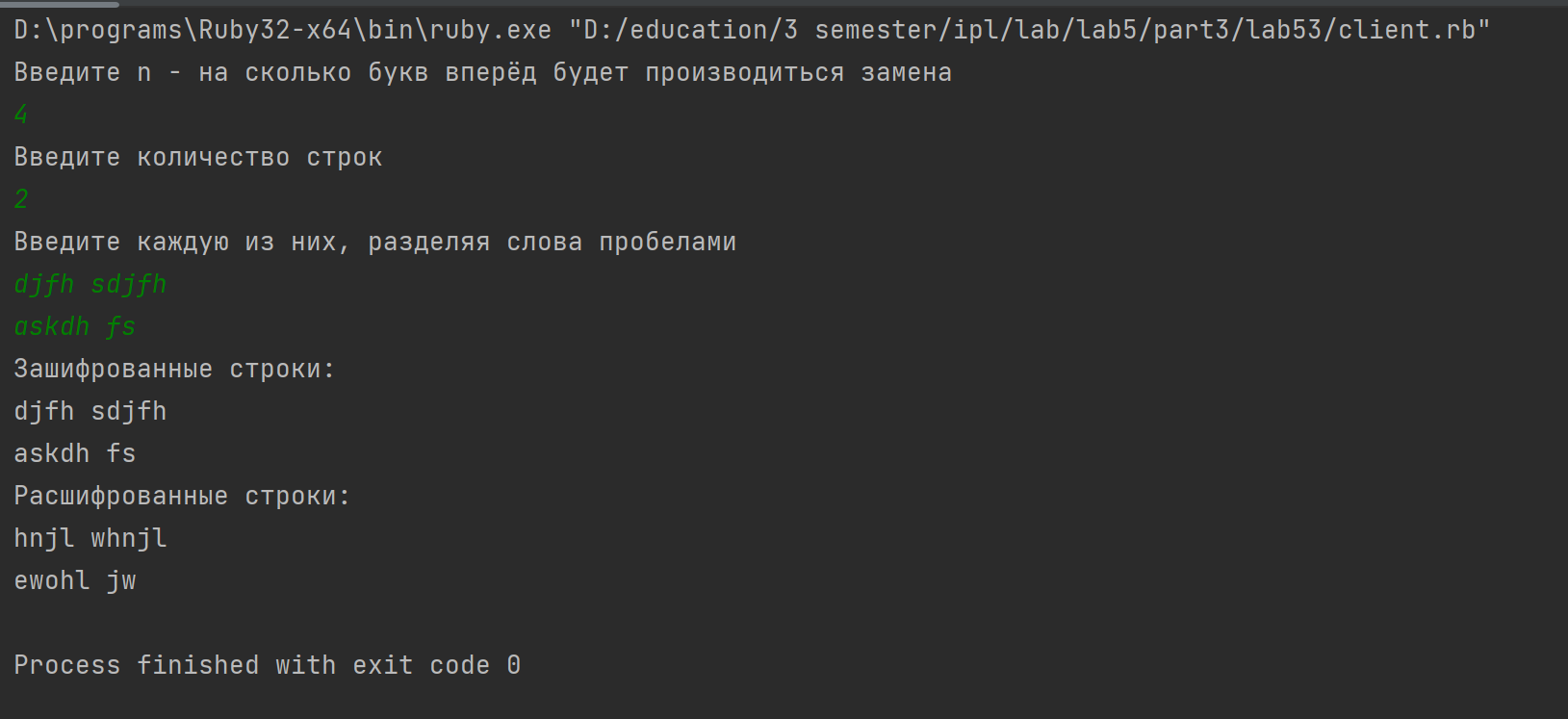
Файл основной программы (main.rb):

*# frozen\_string\_literal: true*def shifr(*str*, *number*)  
 *abc* = ('a'..'z').zip(1..26).to\_h  
 *strmap* = *str*.split  
 *id* = 0 *# индекс слова в строке  
 strmap*.each do |*slovo*|  
 *newstr* = ''  
 *slovo*.each\_char do |*bukva*|  
 *newbukva* = zamenab(*abc*, *bukva*, *number*)  
 *newstr* += *newbukva* end  
 *strmap*[*id*] = *newstr  
 id* += 1  
 end  
 *strmap*.join(' ')  
end  
  
def zamenab(*abc*, *bukva*, *number*)  
 *n* = *abc*[*bukva*].to\_i  
 *n* -= 26 if *n* + *number* > 26  
 *n* += 26 if *n* + *number* < 1  
 *abc*.key(*n* + *number*)  
end

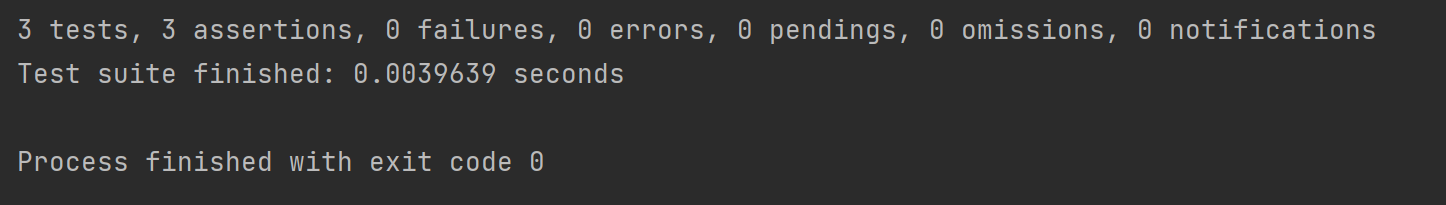
Файл тестов (test.rb):

*# frozen\_string\_literal: true*require 'test/unit'  
require './main'  
  
def random\_str(*len* = 10, *character\_set* = ['a'..'z'])  
 *characters* = *character\_set*.map(&:to\_a).flatten  
 *characters\_len* = *characters*.length  
 (0...*len*).map { *characters*[rand(*characters\_len*)] }.join  
end  
  
def rand\_words(*num*, *len* = 10, *character\_set* = ['a'..'z'])  
 *str* = ''  
 *num*.times do |*\_i*|  
 *str* += "#{random\_str(*len*, *character\_set*)} "  
 end  
 *str*.strip  
end  
  
def prov(*str*, *number*)  
 *str\_map* = *str*.split  
 *newstr* = ''  
 *str\_map*.each do |*slovo*|  
 *newslovo* = zamena(*slovo*, *number*)  
 *newstr* += "#{*newslovo*} "  
 end  
 *newstr*.chop  
end  
  
def zamena(*word*, *number*)  
 *abc* = ('a'..'z').to\_a  
 *new\_abc* = *abc*.zip(newabc(*abc*, *number*)).to\_h  
 *word*.chars.map { |*c*| *new\_abc*.key?(*c*) ? *new\_abc*[*c*] : *c* }.join  
end  
  
def newabc(*abc*, *number*)  
 *newabc* = []  
 *abc*.each\_with\_index do |*\_bukva*, *index*|  
 *number* -= 26 if *index* + *number* > 25  
 *newabc* << (*abc*[*index* + *number*]).to\_s  
 end  
 *newabc*end  
*# str='odunxlklkn ncpjzqomab utmwfnbdjv nyfstfvfix'  
  
# test*class *TestShift* < Test::Unit::TestCase  
 def test\_shift1  
 *n* = 1  
 *str* = rand\_words(4)  
 assert\_equal(prov(*str*, *n*), shifr(*str*, *n*))  
 end  
  
 def test\_shift2  
 *n* = 10  
 *str* = rand\_words(2)  
 assert\_equal(prov(*str*, *n*), shifr(*str*, *n*))  
 end  
  
 def test\_shift3  
 *n* = -1  
 *str* = rand\_words(5)  
 assert\_equal(prov(*str*, *n*), shifr(*str*, *n*))  
 end  
end

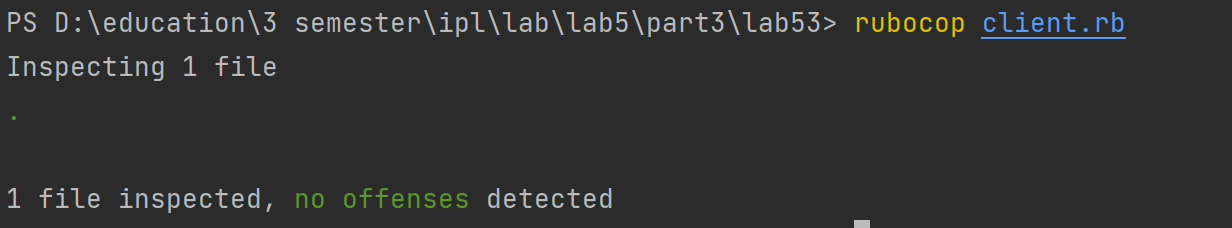
**Результат работы программы:**

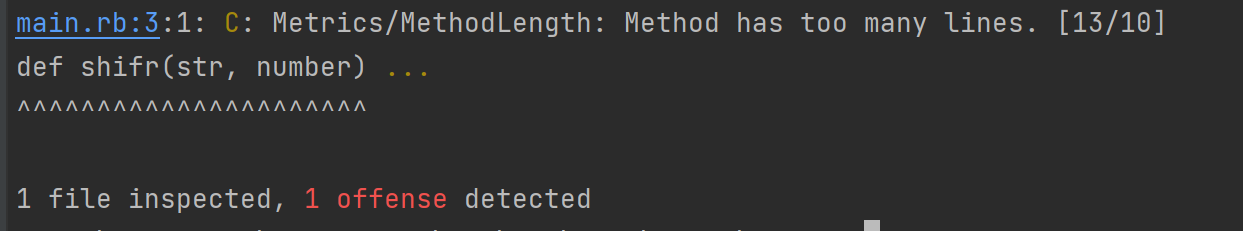


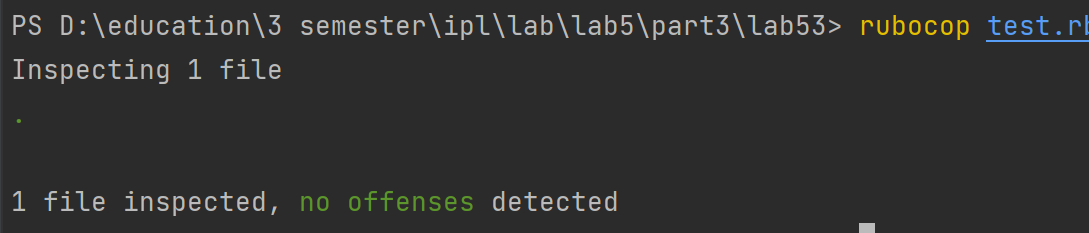
**Результат работы тестов:**



**Результат работы rubocop:**







**Вывод:** мы научились работать с rubocop, создавать тесты и тестировать программу, а также подключать и взаимодействовать с модулями и их методами