**Условие**

Задание выполняется как консольное приложение Ruby. Результат следует предоставлять в виде трех отдельных файлов:

* основная программа;
* программа для взаимодействия с пользователем;
* программа для тестирования на основе

MiniTest::Unit::TestCase

**Тексты программ**

**source.rb**

# This module stores client methods

module Source

def self.change\_grammar\_time(\*args)

result = [] # array with the result string sequence

(0..args.size - 1).each do |count|

str = args[count].split(' ')

result\_str = ''

str.each do |word|

word.gsub!('ing', 'ed') if word.index('ing') == word.size - 3

result\_str += word + ' '

end

result << result\_str

end

result

end

def self.calc\_func(xxx, yyy)

puts 'Source.calc\_func'

xxx = xxx.to\_i

yyy = yyy.to\_i

(2 \* Math.cos(xxx - Math::PI / 6)) / (0.5 + Math.sin(yyy)\*\*2)

end

end

**client.rb**

# frozen\_string\_literal: true

require './source.rb'

# :reek:ManualDispatch

# :reek:UtilityFunction

def execute(method\_name, \*args)

Source.public\_send(method\_name, \*args) if Source.respond\_to? method\_name

end

# :reek:TooManyStatements

def start

puts 'What program do you want to execute?'

p Source.methods - Object.methods

puts "Please, enter a number, corresponding to the func position in the list.

Then enter arguments, if needed (type 'stop' to stop):"

response = gets.chomp.to\_i

puts 'enter arguments, if needed'

arguments = gets('stop').chomp('stop')

[response, arguments]

end

start\_response = start

response = start\_response[0]

arguments = start\_response[1]

method\_names = (Source.methods - Object.methods)

MIN\_POSITION = 1

if response && response >= MIN\_POSITION && response <= method\_names.size

name = method\_names[response.to\_i - 1]

formatted\_arguments = arguments.split("\n")

case formatted\_arguments.size

when 0

execute(name)

when 1

execute(name, formatted\_arguments[0])

when 2

execute(name, formatted\_arguments[0],

formatted\_arguments[1])

when 3

execute(name,

formatted\_arguments[0],

formatted\_arguments[1],

formatted\_arguments[2])

when 4

execute(name, formatted\_arguments[0],

formatted\_arguments[1],

formatted\_arguments[2],

formatted\_arguments[3])

else

puts 'Too many arguments'

end

end

**test.rb**

# frozen\_string\_literal: true

require\_relative 'source.rb'

require 'minitest/autorun'

def generate\_str\_sequence(number\_of\_times)

random\_input = []

random\_result = []

number\_of\_times.times do

random\_str = generate\_string

random\_input << random\_str[1]

random\_result << random\_str[0]

end

[random\_input, random\_result]

end

def generate\_string

random\_string = ''

result\_string = ''

5.times do

random\_word\_base = (0...8).map { rand(97..122).chr }.join

random\_string += random\_word\_base

result\_string += random\_word\_base

if rand(0..1) == 1

random\_string += 'ing'

result\_string += 'ed'

end

random\_string += ' '

result\_string += ' '

end

[result\_string, random\_string]

end

# Class for setting any tests on functions

class Tests < MiniTest::Test

def setup; end

def test\_calc\_func

assert\_equal(-1.4084366469497145, Source.calc\_func(41, 2))

assert\_equal(-0.32940836640837823, Source.calc\_func(5, 5))

end

# :reek:FeatureEnvy

def test\_change\_grammar\_time

invoke = generate\_str\_sequence(3)

input\_strings = invoke[0]

assert\_equal(invoke[1], Source.change\_grammar\_time(input\_strings[0],

input\_strings[1],

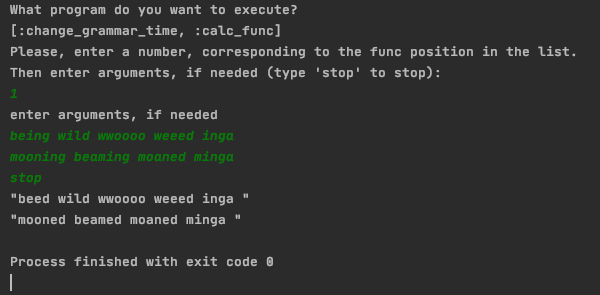
input\_strings[2]))

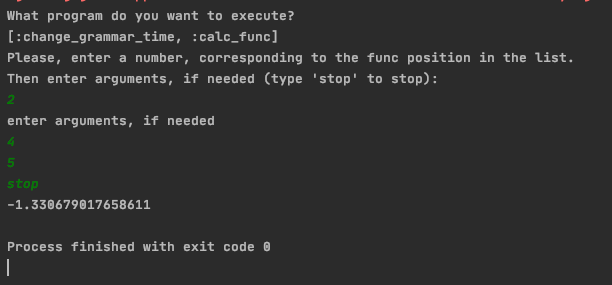
end

def teardown; end

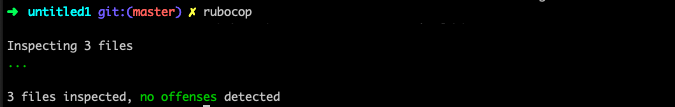
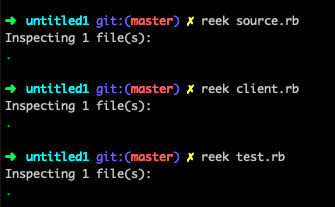
end

**Результаты выполнения**

****

****

**Результаты проверки анализаторами rubocop и reek**

****