|  |  |
| --- | --- |
| **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 |

Название: Программирование на языке Ruby

**Дисциплина:** Языки интернет-программирования

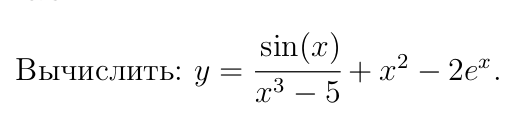
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-35Б |  |  | Л.Э. Барсегян |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

Вариант № 5.02

**Часть 1.**

Задание:



**Код программы.**

**Код файла calc.rb**

# frozen\_string\_literal: true

# module Calculate

module Calculate

def self.calc(input)

return 'InputError: expected a numeric input' if input == '' || !input.match?(/^-?[0-9]\d\*(\.\d+)?$/)

@input = input.to\_f

(Math.sin(@input) / (@input\*\*3 - 5) + (@input\*\*2) - 2 \* Math.exp(@input)).round(2)

end

end

**Код файла main.rb**

# frozen\_string\_literal: true

require\_relative 'calc'

# module Main

module Main

include Calculate

def self.main

print 'Введите x: '

input = gets.chop

puts 'Результат вычислений: ' unless Calculate.calc(input).is\_a?(String)

puts Calculate.calc(input)

end

end

Main.main

**Код файла calc\_spec.rb**

# frozen\_string\_literal: true

require\_relative 'calc'

require 'faker'

RSpec.describe Calculate do

include Calculate

include Faker

describe '#calc' do

context 'when parameter is empty' do

it 'raises an error' do

expect(Calculate.calc('')).to eq('InputError: expected a numeric input')

end

end

context 'when parameter is a valid string' do

it 'contains not numeric characters' do

test\_strings = ['3e', 'qwe', '12qwe', '3!3', '-3e', '-', ' ', '']

test\_strings.each do |test\_string|

expect(Calculate.calc(test\_string)).to eq('InputError: expected a numeric input')

end

end

it 'is numeric' do

nums = [-3, 0, 10, 13]

answers = [8.90, -2, -43\_952.93, -884\_657.78]

nums.each\_index { |i| expect(Calculate.calc(nums[i].to\_s)).to eq(answers[i]) }

end

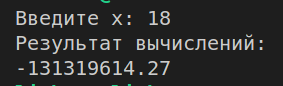
end

end

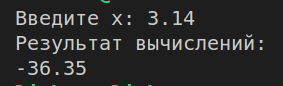
end

**Результаты выполнения программы**

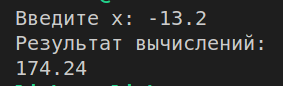
**1. Ввод целого числа**

****

**2. Ввод десятичного числа**

****

**3. Ввод отрицательного числа**

****

**4. Ввод недопустимых значений**

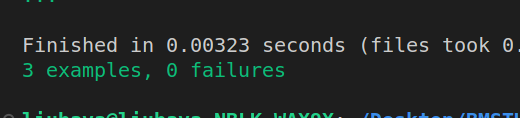
****

****

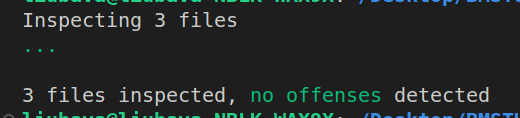
****

****

**Результат тестирования на основе RSpec**

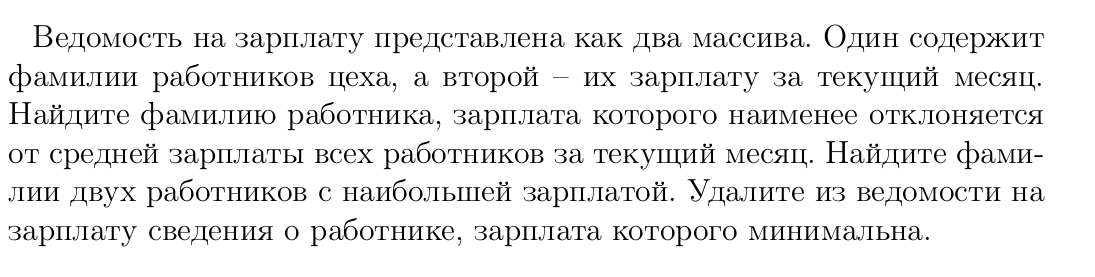
****

**Проверка на соответствие стилю программирования на Ruby при помощи rubocop**

****

**Часть 2.**

**Задание:**

****

**Код файла main.rb**

**# frozen\_string\_literal: true**

require\_relative 'arrays'

# module Main

module Main

include Arrays

@workers = []

@salaries = []

def self.eol?(input)

input == ''

end

def self.input\_worker

print "\nДля остановки ввода нажмите Enter\n\nВведите фамилию работника: "

@worker = gets.chomp

return unless eol?(@worker)

print "\nКонец ввода.\n"

-1

end

def self.input\_salary

print "Введите зарплату за текущий месяц работника #{@worker}: "

@salary = gets.chomp

if eol?(@salary)

print "\nКонец ввода.\n"

return -1

end

return if @salary.match?(/^[.0-9]\*$/)

print "\nОшибка ввода.\n"

-1

end

def self.input\_data

loop do

return -1 if input\_worker == -1

return -1 if input\_salary == -1

@workers.append(@worker)

@salary = @salary.to\_f

@salary = @salary.to\_i if @salary.to\_f.to\_s == @salary.to\_i.to\_f.to\_s

@salaries.append(@salary.to\_i)

end

end

def self.main

return if input\_data == -1 && @workers.size.zero?

Arrays.print\_info(@workers, @salaries)

Arrays.print\_av(@workers, @salaries)

Arrays.print\_richest(@workers, @salaries)

Arrays.print\_min\_dev(@workers, @salaries)

Arrays.fire\_loser(@workers, @salaries)

Arrays.print\_info(@workers, @salaries)

end

end

Main.main

**Код файла arrays.rb**

**# frozen\_string\_literal: true**

# module Arrays

module Arrays

@two\_richest = []

attr\_reader :workers

attr\_writer :salaries

def self.find\_av\_salary(\_workers, salaries)

return if salaries.size.zero?

(salaries.sum.to\_f / salaries.size).round(3)

end

def self.find\_richest(workers, salaries)

return if salaries.size != workers.size || workers.size.zero?

@two\_richest.append(workers[salaries.index(salaries.max(2)[0]).to\_i])

@two\_richest.append(workers[salaries.index(salaries.max(2)[1]).to\_i])

end

def self.find\_min\_deviation(workers, salaries)

return if salaries.size != workers.size || workers.size.zero? || salaries.size.zero?

workers[salaries.index(salaries.min\_by { |x| (find\_av\_salary(workers, salaries) - x).abs })]

end

def self.print\_info(workers, salaries)

puts "Информация о всех работниках:\n"

workers.each\_index do |i|

puts "#{i + 1}: #{workers[i]} #{salaries[i]}"

end

end

def self.print\_av(workers, salaries)

puts "Средняя зарплата всех работников за текущий месяц: #{find\_av\_salary(workers, salaries)}"

end

def self.print\_richest(workers, salaries)

puts "Работники с наибольшей зарплатой: #{find\_richest(workers,

salaries)[0]} и #{find\_richest(workers, salaries)[1]}"

end

def self.print\_min\_dev(workers, salaries)

puts "Работник, зарплата которого наименее отклоняется от средней зарплаты всех работников: #{find\_min\_deviation(

workers, salaries

)}"

end

def self.fire\_loser(workers, salaries)

puts 'Поиск и увольнение работника с минимальной зарплатой.'

workers.delete\_at(salaries.index(salaries.min))

salaries.delete\_at(salaries.index(salaries.min))

end

end

**Код файла arrays\_spec.rb**

# frozen\_string\_literal: true

require\_relative 'arrays'

RSpec.describe Arrays do

include Arrays

describe '#find\_av\_salary' do

context 'when salaries array is empty' do

it 'must return nil' do

expect(Arrays.find\_av\_salary([], [])).to eq(nil)

end

end

context 'when arrays are not empty' do

workers = %w[Petrov Ivanov Smirnov Sidorov]

salaries = [43\_000, 260\_000, 124\_000, 13\_000]

it 'must count correctly' do

expect(Arrays.find\_av\_salary(workers, salaries)).to eq(110\_000)

end

end

end

end

RSpec.describe Arrays do

include Arrays

describe '#find\_richest' do

context 'when arrays are empty' do

it 'must return nil' do

expect(Arrays.find\_richest([], [])).to eq(nil)

end

end

context 'when one of the arrays is is empty' do

it 'must return nil' do

expect(Arrays.find\_richest([], [30, 50])).to eq(nil)

end

end

context 'when arrays have different size' do

it 'must return nil' do

expect(Arrays.find\_richest(%w[Ivanov Petrov], [45\_000])).to eq(nil)

end

end

context 'when arrays are valid' do

it 'must work correctly' do

expect(Arrays.find\_richest(%w[Petrov Ivanov Smirnov Sidorov],

[43\_000, 260\_000, 124\_000, 13\_000])).to eq(%w[Ivanov Smirnov])

end

end

end

end

RSpec.describe Arrays do

include Arrays

describe '#find\_min\_deviation' do

context 'when arrays are empty' do

it 'must return nil' do

expect(Arrays.find\_min\_deviation([], [])).to eq(nil)

end

end

context 'when one of the arrays is is empty' do

it 'must return nil' do

expect(Arrays.find\_min\_deviation([], [30, 50])).to eq(nil)

end

end

context 'when arrays have different size' do

it 'must return nil' do

expect(Arrays.find\_min\_deviation(%w[Ivanov Petrov], [45\_000])).to eq(nil)

end

end

context 'when arrays are valid' do

it 'must work correctly' do

expect(Arrays.find\_min\_deviation(%w[Petrov Ivanov Smirnov Sidorov],

[43\_000, 260\_000, 124\_000, 13\_000])).to eq('Smirnov')

end

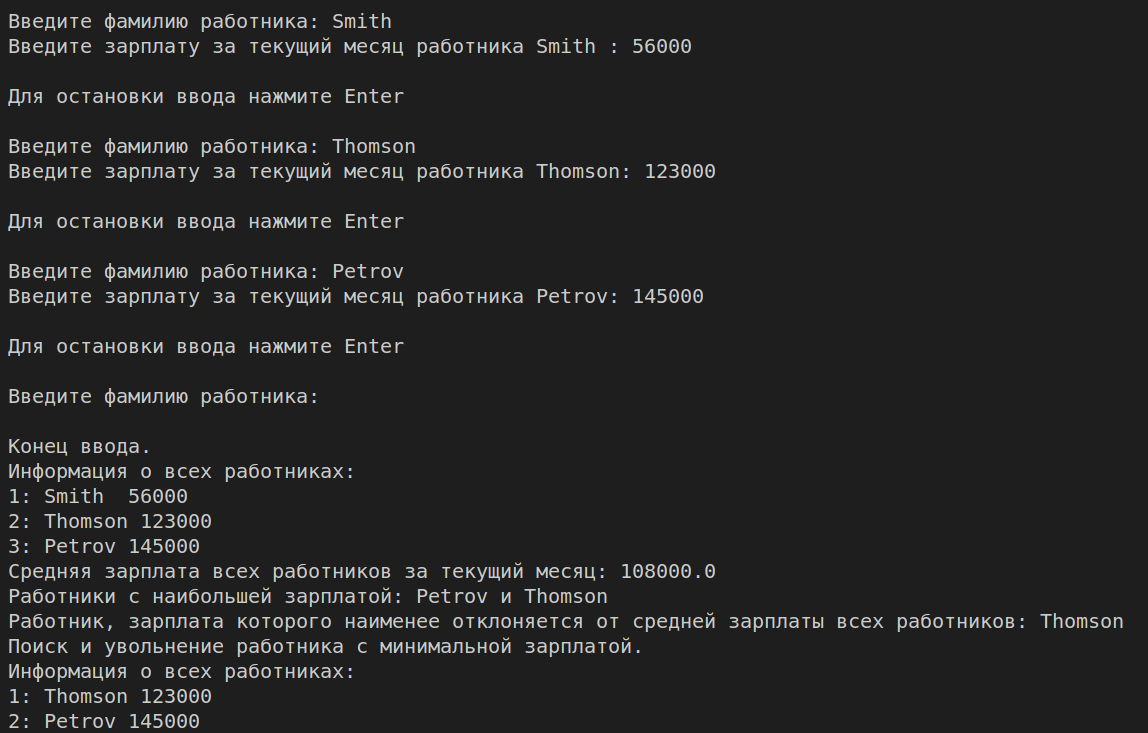
end

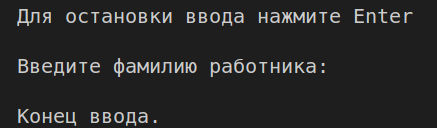
end

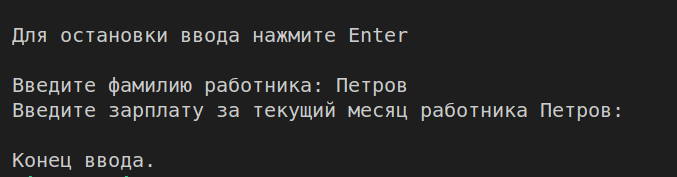
end

**Результаты выполнения программы**

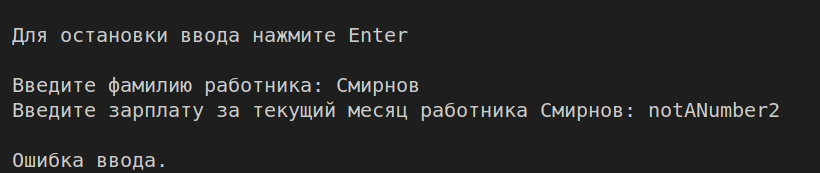
**1. Ввод валидных данных**

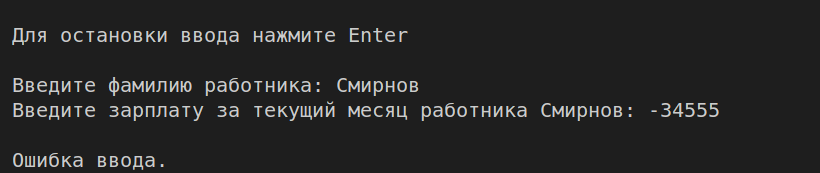
**2. Нажатие Enter**

****

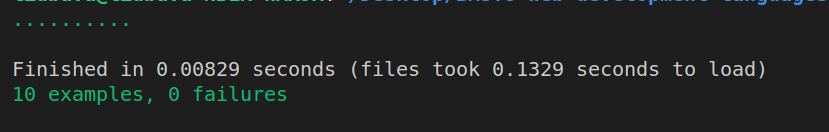
****

**3. Ввод недопустимых значений**

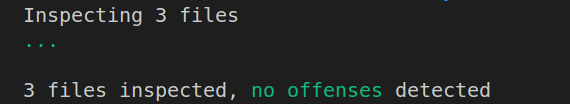
****

****

**Результат тестирования на основе Rspec**

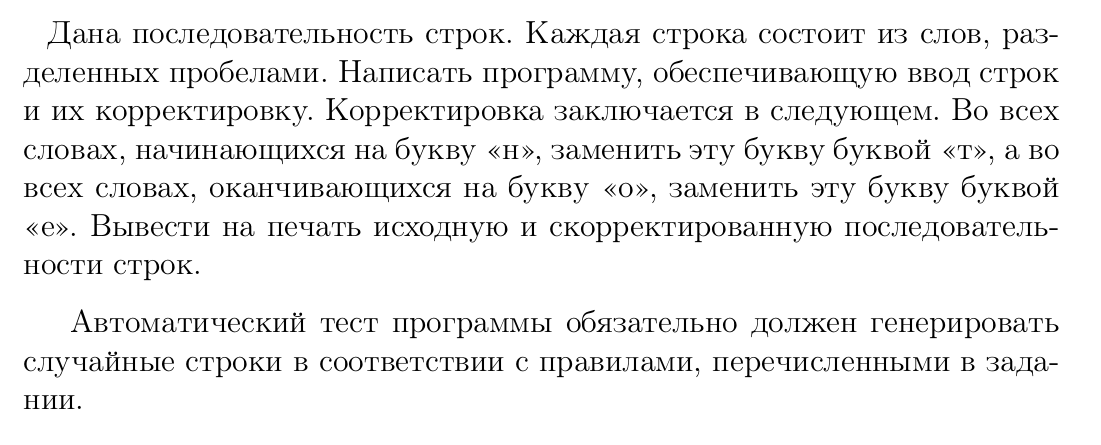
****

**Проверка на соответствие стилю программирования на Ruby при помощи rubocop**

****

**Часть 3.**

**Задание:**

****

**Код файла main.rb**

# frozen\_string\_literal: false

require\_relative 'strings'

# module IOcontroller

module IOcontroller

include Strings

def self.eol?(string)

string == ''

end

def self.input\_strings

strings = []

print "\nВведите Enter для остановки ввода.\n\n"

puts 'Введите последовательность строк, состоящих разделенных пробелами строк'

loop do

string = gets.chomp

break if eol?(string)

strings.append(string)

end

strings

end

def self.main

strings = input\_strings

print "Строки до исправления:\n"

Strings.print\_strings(strings)

Strings.correct\_strings(strings)

print "\nСтроки после исправления:\n"

Strings.print\_strings(strings)

end

end

IOcontroller.main

**Код файла strings.rb**

# frozen\_string\_literal: true

# module Strings

module Strings

def self.print\_strings(strings)

puts strings

end

def self.correct\_strings(strings)

return if strings.empty?

strings.each do |str|

str.gsub!(/^[Нн]/, 'т')

str.gsub!(/ [Нн]/, ' т')

str.gsub!(/[оО]$/, 'е')

str.gsub!(/[оО] /, 'е ')

end

end

end

**Код файла strings\_spec.rb**

# frozen\_string\_literal: false

require\_relative 'strings'

RSpec.describe Strings do

include Strings

describe '#correct\_strings' do

context 'when the string is valid' do

it 'corrects the string' do

strings = ['Наша Таня громко плачет', 'Уронила в речку мячик']

expect(Strings.correct\_strings(strings)).to eq(['таша Таня громке плачет', 'Уронила в речку мячик'])

end

end

context 'when the array of strings is empty' do

it 'returns nil' do

expect(Strings.correct\_strings([])).to eq(nil)

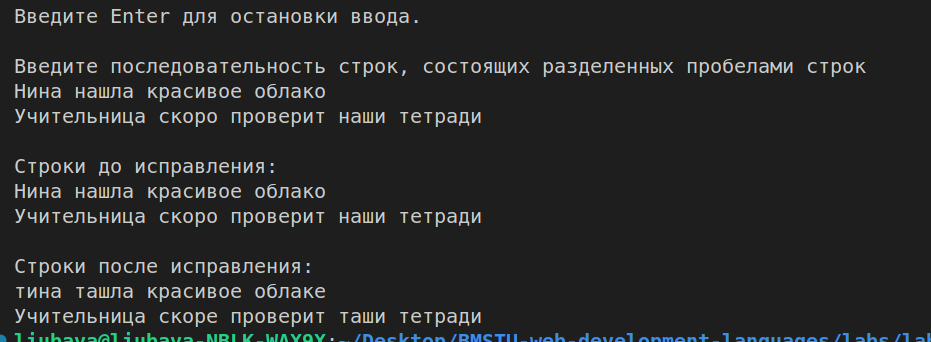
end

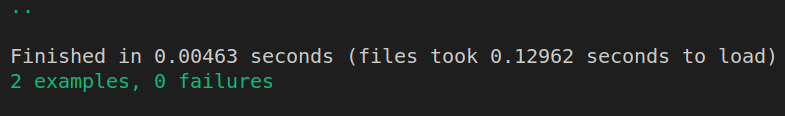
end

end

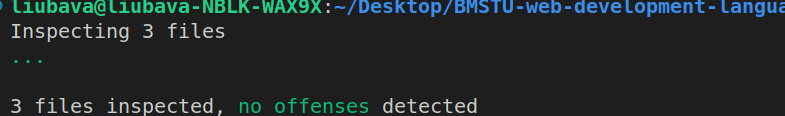
end

**Результаты выполнения программы**

**Резу****льтат тестирования на основе Rspec**

****

**Проверка на соответствие стилю программирования на Ruby при помощи rubocop**

**Вывод:** изучила основы синтаксиса Ruby, научилась писать консольные приложения с вводом пользователем данных