|  |  |
| --- | --- |
| **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:**

Программы на Ruby без использования циклов

for, while, loop. Обработка массивов.

**Дисциплина:** Языки Интернет-программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-34Б |  | 15.10.2022 | С. А. Рахманов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Д. В. Малахов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

**Цель работы:** получение навыков программирования на языке Ruby с использованием функционального стиля программирования, ознакомиться со средствами проверки соответствия стиля программирования.

**Задание:**

Все консольные приложения Ruby следует реализовывать в виде трех

отдельных файлов:

1. основная программа;

2. программа для взаимодействия с пользователем через консоль;

3. программа для автоматического тестирования на основе MiniTest::Unit

или RSpec. Везде, где это возможно, данные для проверки должны

формироваться автоматически по правилам, указанным в задании.

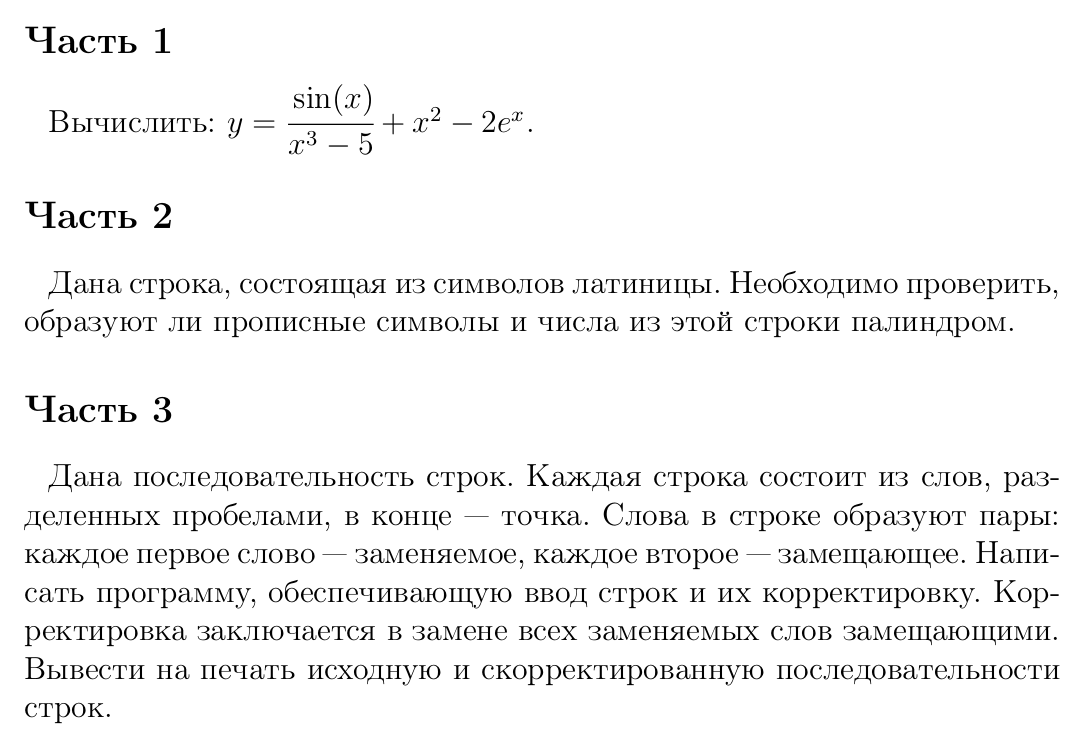
При реализации программ везде, где это возможно, следует избегать

использования циклов for, do, while. Вместо них используйте методы из

примеси Enumerable.

Все тексты программ должны быть проверены на соответствие стилю

программирования Ruby при помощи rubocop.ru или reek.



**Тексты программ**

**Часть 1**

*interface.rb*

# frozen\_string\_literal: true

require './main.rb'

loop do

print 'Enter X: '

x = gets.to\_f

print "Y = #{calc(x).round(4)}"

puts

end

*main.rb*

# frozen\_string\_literal: true

def calc(number)

Math.sin(number) / (number\*\*3 - 5) + number\*\*2 - 2 \* Math.exp(number)

end

*test.rb*

# frozen\_string\_literal: true

require 'minitest/autorun'

require './main.rb'

# Test Class

class TestTree < Minitest::Test

# first test

def test\_1

data = [ # [x, res(y)]

[0, -2],

[1, -4.6469],

[2, -10.475],

[-1, 0.4045]

]

# data = Array.new(5) {rand(-10..10)}

data.each do |d|

res = calc(d[0])

assert\_in\_delta(d[1], res, 0.0001, ["Incorrect value function calc(#{d[0]}) -> #{d[1]} (вместо #{res})"])

end

end

end

**Часть 2**

*interface.rb*

# frozen\_string\_literal: true

require './main.rb'

loop do

print 'Enter string: '

str = gets

tmp = palindrom?(str)

# p isPalindrom?(str)

print "Symbols and numbers is palindrom - #{tmp}"

puts

end

*main.rb*

# frozen\_string\_literal: true

def palindrom?(str)

str = str.scan(/\d|[a-z]/).join

# p str

str == str.reverse && !str.empty?

end

*test.rb*

# frozen\_string\_literal: true

require 'minitest/autorun'

require './main.rb'

# Test Class

class TestTree < Minitest::Test

# first test

def test\_1

data = [ # [str, str.reverse]

['teLOLnet', true], # tenet

['1r oRRR =tator1', true], # 1rotator1

['k aDya k', true], # kayak

['wo w', true], # wow

['1>:D23W21', true], # 12321

['troll', false]

]

data.each do |d|

res = palindrom?(d[0])

assert\_equal(res, d[1], ['Correct work of function "isPalindrom?"'])

end

end

end

**Часть 3**

*interface.rb*

# frozen\_string\_literal: true

require './main.rb'

loop do

print 'Enter subsequence of strings:'

str = gets.chop

print correction(str)

puts

end

*main.rb*

# frozen\_string\_literal: true

def correction(str)

new\_str = ''

str.split('.').each do |sentence|

words = sentence.split(' ')

words.each\_index do |index|

if index % 2 == 0

words[index] = words[index + 1]

end

end

# words.map\_with\_index { |val, index| (index.even? ? words[index + 1] : val)}

words\_str = words.join(' ')

new\_str += words\_str + '. '

end

new\_str.chop

end

*test.rb*

# frozen\_string\_literal: true

require 'minitest/autorun'

require './main.rb'

# Test Class

class TestTree < Minitest::Test

# first test

def test\_1

data = [

['1 2 3 4. 1 2 3 4 5 6.', '2 2 4 4. 2 2 4 4 6 6.'],

['1 2 3 4 7 8. 1 2 3 4 5 6. 1 2. 1 2.', '2 2 4 4 8 8. 2 2 4 4 6 6. 2 2. 2 2.'],

['It is my mistake. Sorry i need to go out.', 'is is mistake mistake. i i to to out out.'],

['ab ba. cd dc. ef fe. gh hg ij ji kl lk mn nm.', 'ba ba. dc dc. fe fe. hg hg ji ji lk lk nm nm.'],

['BA AB. DC CD. FE EF. HG GH JI IJ LK KL NM MN', 'AB AB. CD CD. EF EF. GH GH IJ IJ KL KL MN MN.']

]

data.each do |elem|

temp\_str = correction(elem[0])

assert\_equal(temp\_str, elem[1], 'Incorrect work of function "correction()"')

end

end

# def generateStr

# rand(65..90).chr \* rand(5..10)

# end

# def test\_auto

# 10.times do

# temp\_str = generateStr()

# cor\_temp\_str = correction(temp\_str)

# assert\_equal(t1, "#{name} #{otchestvo} #{surename}", ['Incorrect value function "correction"'])

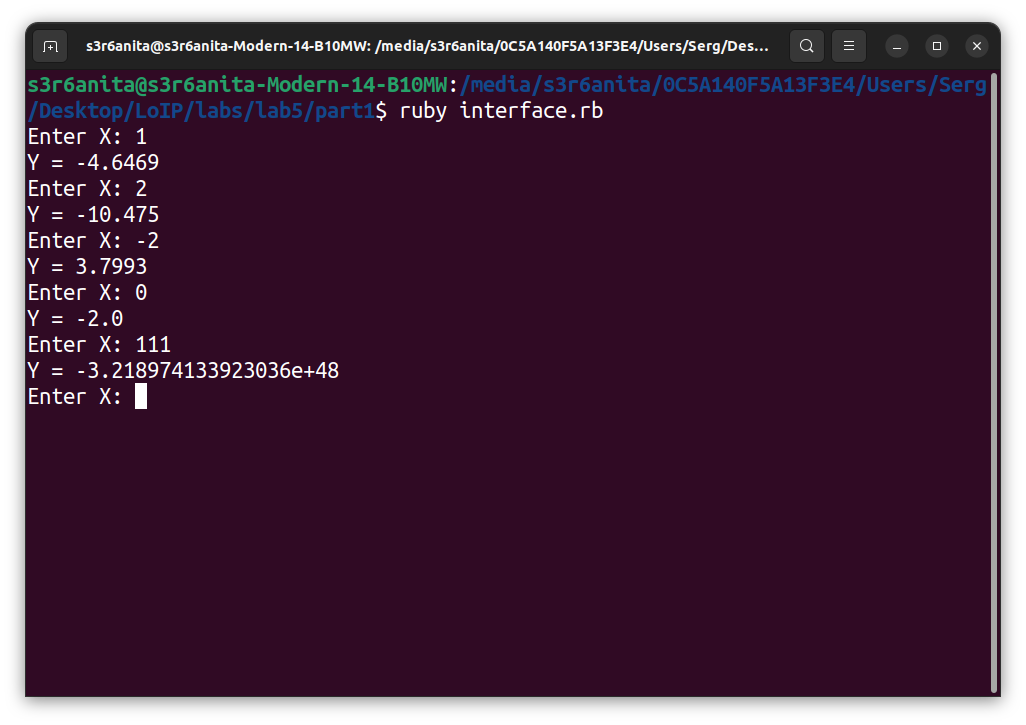
# end

# end

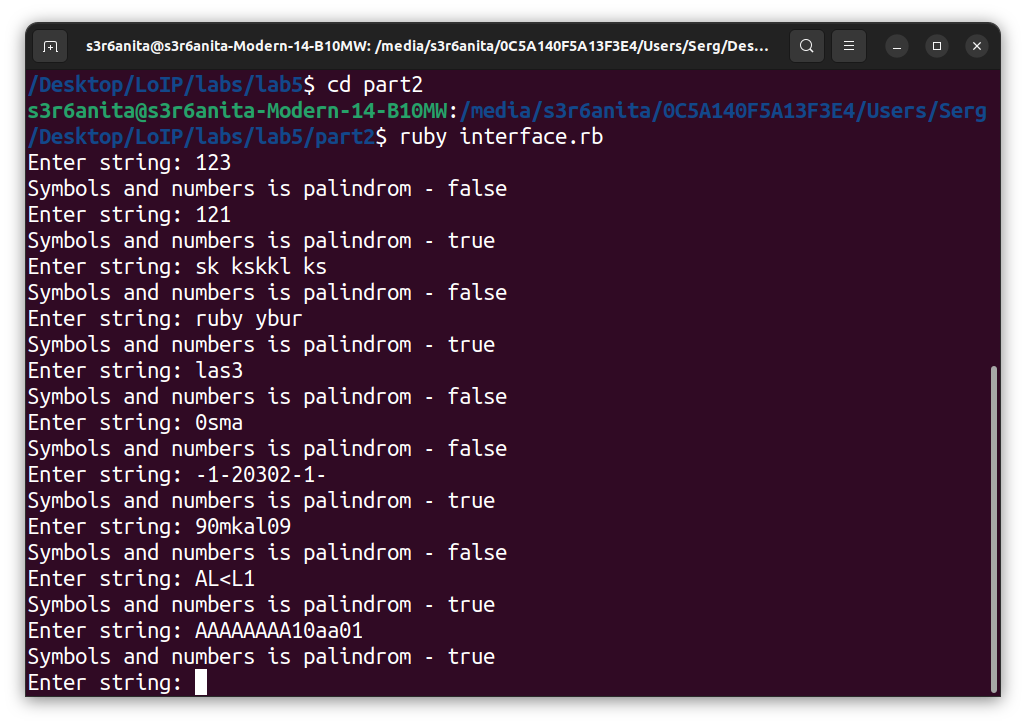
end

**Результаты выполнения**

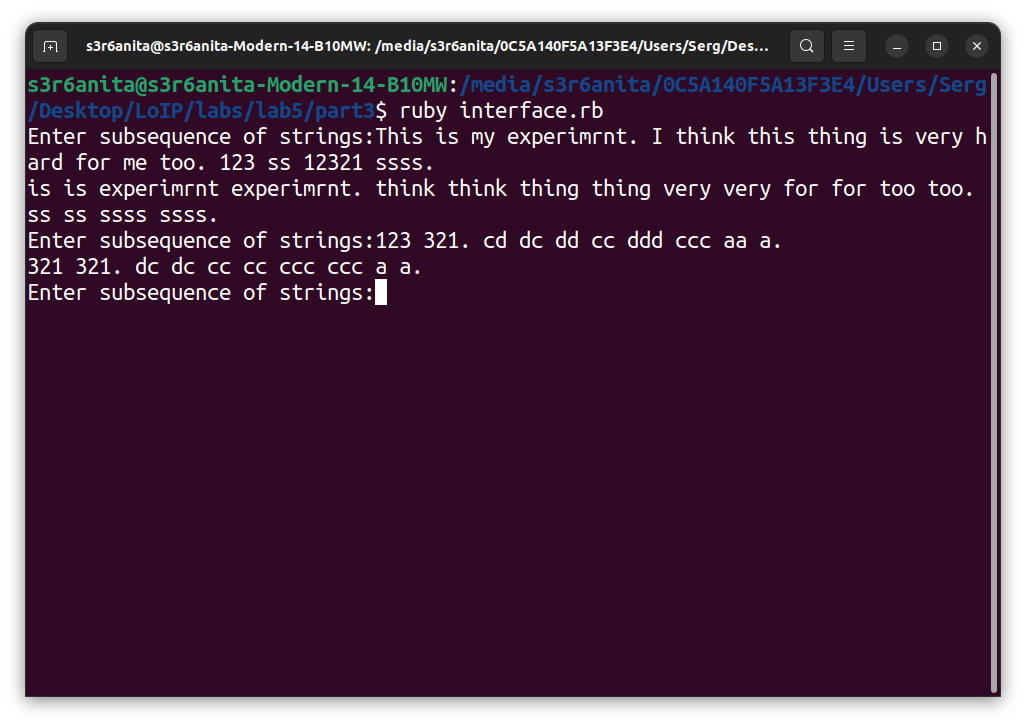
**Часть 1**

****

**Часть 2**

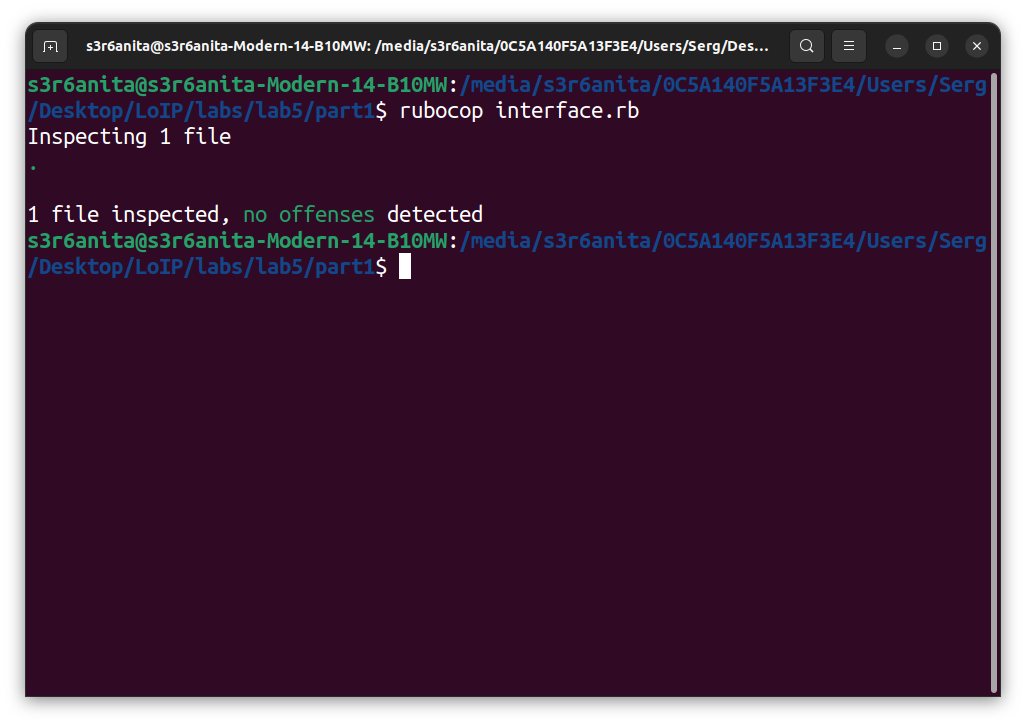
****

**Часть 3**

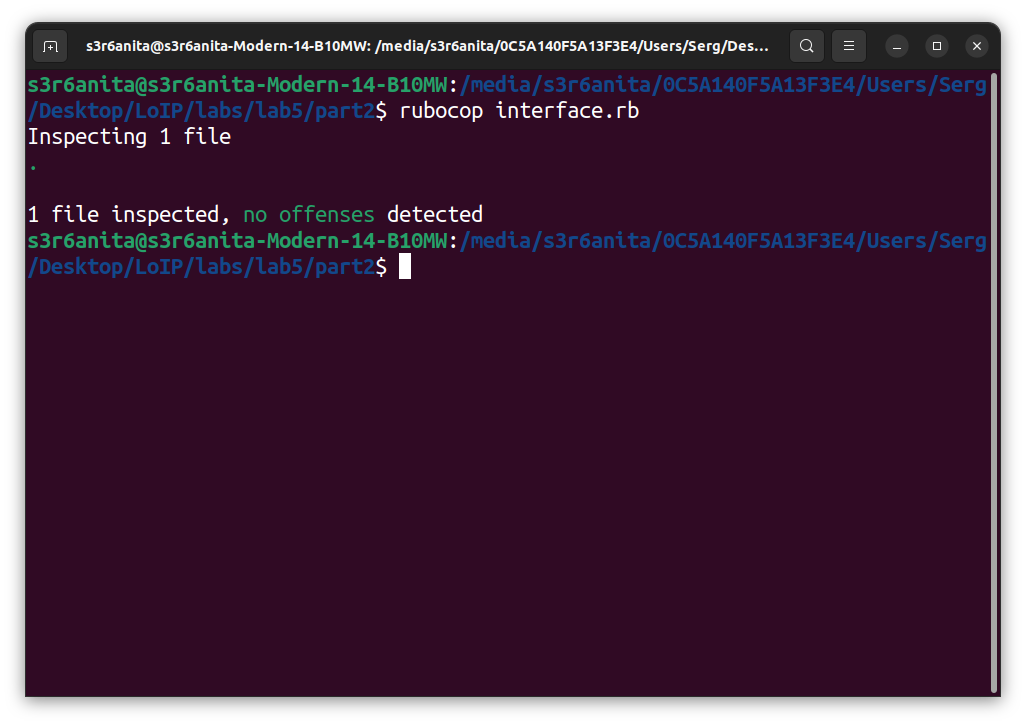
****

**Результаты проверки анализатором rubocop**

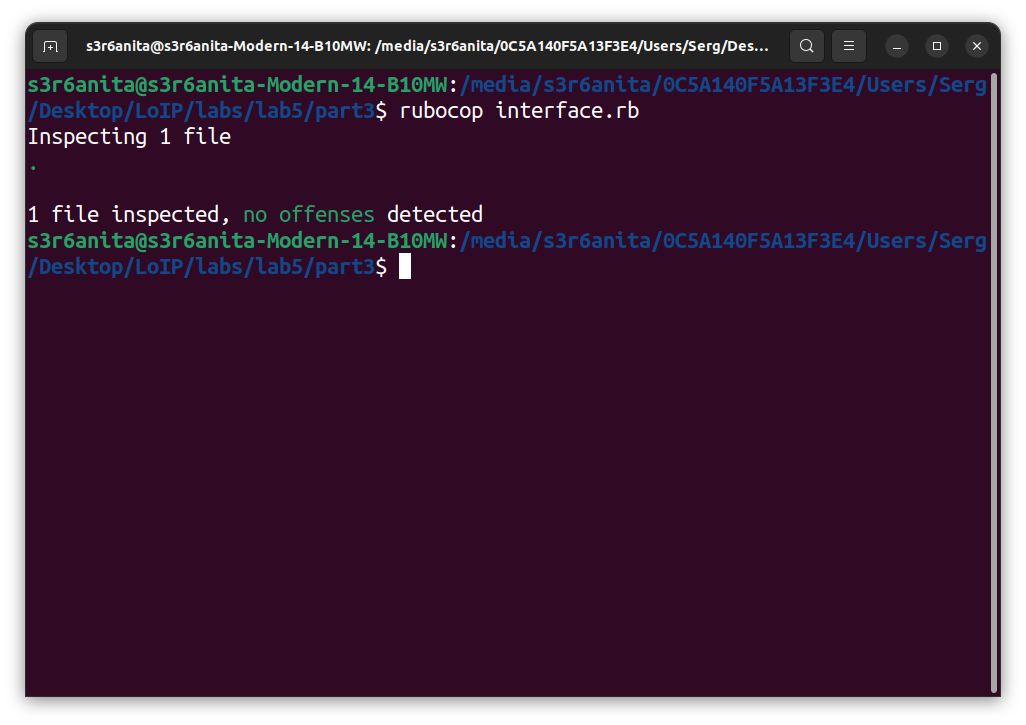
**Часть 1**

****

**Часть 2**

****

**Часть 3**

****

**Вывод:** получил навыки программирования на языке Ruby с использованием функционального стиля программирования, получил навыки использования rubocop и проверил написанный код на соответствие стилю программирования на Ruby.