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 – 09.03.01 Информатика и вычислительная техника

О Т Ч Е Т

по лабораторной работе **№** 6

Название: Консольные приложения Ruby

Дисциплина: Языки интернет-программирования

Студент ИУ6-32Б _____ Джафаров Э.Э.
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Петрова Я.С.
(Подпись, дата) (И.О. Фамилия)

Москва, 2022

Цель работы:

Целью работы является ознакомление с языком Ruby и написании программ и тестов

Задание:

Часть 1

Решить задачу, организовав итерационный цикл с точностью ξ = 10^−4 , 10^−5 .
Вычислить сумму ряда:

$$S = \sum_{k=1}^{\infty} \frac{1}{k(k+1)},$$

, точное значение равно 1.
Определить, как изменяется число итераций при изменении точности.

Часть 2

Решить предыдущее задание с помощью Enumerable или Enumerator.

Часть 3

Составить метод trap для вычисления определенного интеграла по формуле трапеций

$$\int_a^b f(x)\,\mathrm{d}x \approx \left(\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(x_i)\right) \cdot \frac{b-a}{n},$$

где f (x) подынтегральная функция, [a, b] - интервал интегрирования, n - число отрезков разбиения. В основной программе использовать метод trap для вычисления интегралов:

$$\int_{-1}^{4} (x+\cos x)\,\mathrm{d}x \text{ и } \int_1^2 \frac{tg(x+1)}{x+1}\,\mathrm{d}x.$$

Реализовать вызов метода двумя способами: в виде передаваемого lambda-выражения и в виде блока.

Решение:

Часть 1

1. Основная программа

```ruby
# frozen_string_literal: true

# Class for Ryad
class Ryad
  def initialize(power)
    @summa = 0
    @k = 1.0
    @e = 1 / 10.0**power
    @slag = 0
  end

  def count
    100_000_000_000.times do
      @slag = 1 / (@k * (@k + 1))
      @summa += @slag
      @k += 1.0
      break if @summa > 1 - @e
    end
  end

  def get
    @summa
  end

  def prt
    puts "Количество итераций: #{(@k - 1).to_i}"
    puts "Точность: #{@e}"
    puts "Сумма: #{@summa}"
  end
end
```

2. Программа для взаимодействия с пользователем через консоль

```ruby
# frozen_string_literal: true

# get input
require_relative 'ryad'

puts 'Enter 4 or 5'
a = Ryad.new(gets.to_i)
a.count
a.prt

```

3. Тест

```ruby
# frozen_string_literal: true

# test
require_relative 'ryad'

RSpec.describe Ryad do
  it 'must return right value with accuracy 10^(-4)' do
    a = Ryad.new(4)
    a.count
    expect(a.get).to be_within(0.0001).of(1)
  end

  it 'must return right value with accuracy 10^(-4)' do
    a = Ryad.new(5)
    a.count
    expect(a.get).to be_within(0.00001).of(1)
  end
end

```

4. Результат

1.1 Основной программы

```
Enter 4 or 5
6
Количество итераций: 999999
Точность: 1.0e-06
Сумма: 0.9999990000000476
```

1.2 Теста

```
..

Finished in 0.10329 seconds (files took 1.6 seconds to load)
2 examples, 0 failures
```

5. Проверка Rubocop:

```
Inspecting 3 files
...

3 files inspected, no offenses detected
```

6. Проверка Reek:

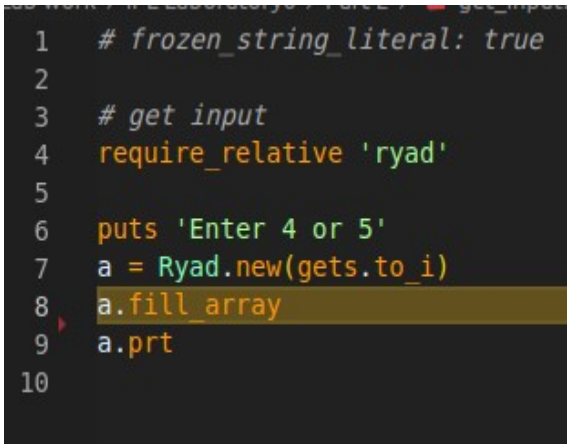
```
rk/IPL-Laboratory6$ reek Part\ 1
Inspecting 3 file(s):
...

0 total warnings
```

Часть 2

1. Основная программа

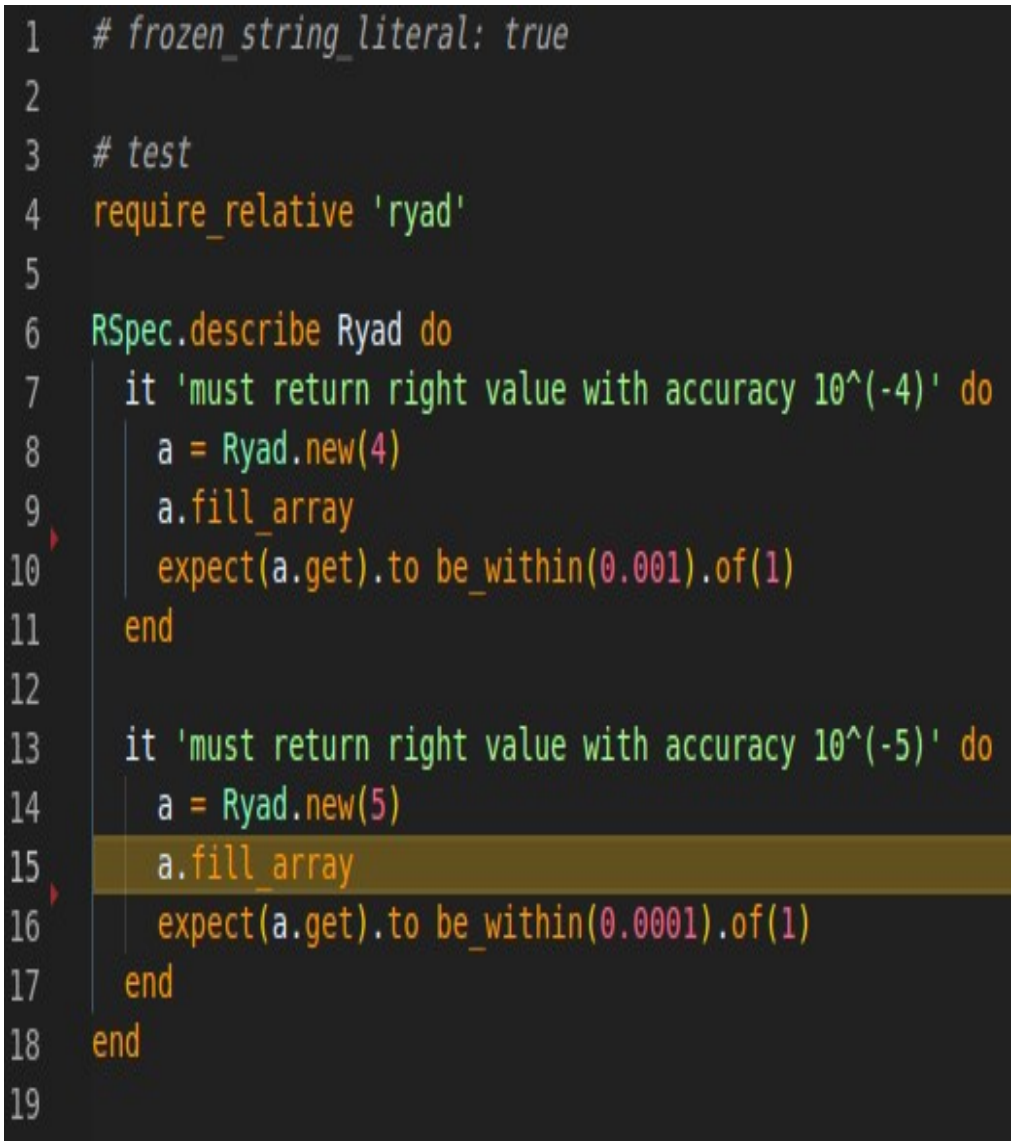
```ruby
# frozen_string_literal: true

# Class for Ryad with Enumerable
class Ryad
  def initialize(power)
    @summa = 0
    @k = 1.0
    @e = 1 / 10.0**power
    @slag = 0
  end

  def fill_array
    @enu = []
    @i = 0
    100_000_000_000.times do
      @slag = 1 / (@k * (@k + 1))
      @enu[@i] = @slag
      @i += 1
      @k += 1.0
      @summa += @slag
      return @summa if @summa + @e > 1.0
    end
  end

  def count
    @k = 0
    @summa = 0
    @summa = @enu.inject(0, :+)
    puts @summa
  end

  def get
    @summa
  end

  def prt
    puts "Количество итераций: #{(@i - 1).to_i}"
    puts "Точность: #{@e}"
    puts "Сумма: #{@summa}"
  end
end

```

2. Программа для взаимодействия с пользователем через консоль



```ruby
# frozen_string_literal: true

# get input
require_relative 'ryad'

puts 'Enter 4 or 5'
a = Ryad.new(gets.to_i)
a.fill_array
a.prt

```

3. Тест



```ruby
# frozen_string_literal: true

# test
require_relative 'ryad'

RSpec.describe Ryad do
  it 'must return right value with accuracy 10^(-4)' do
    a = Ryad.new(4)
    a.fill_array
    expect(a.get).to be_within(0.001).of(1)
  end

  it 'must return right value with accuracy 10^(-5)' do
    a = Ryad.new(5)
    a.fill_array
    expect(a.get).to be_within(0.0001).of(1)
  end
end

```

4. Результат

1.1 Основной программы

```
Enter 4 or 5
7
Количество итераций: 10000018
Точность: 1.0e-07
Сумма: 0.9999999000000052
```

1.2 Теста

```
..

Finished in 0.12817 seconds (files took 0.28522 seconds to load)
2 examples, 0 failures
```

5. Проверка Rubocop:

```
Inspecting 3 files
...

3 files inspected, no offenses detected
```

6. Проверка Reek:

```
rk/IPL-Laboratory6$ reek Part\ 2/
Inspecting 3 file(s):
...

0 total warnings
```

Часть 3

1. Основная программа

```ruby
# frozen_string_literal: true

# Calculate func
class Calculator
  def initialize(down, upper)
    @a = down
    @b = upper
    @x = 1
    @n = (@a.abs + @b.abs)
    @summa = 0
  end

  def calc_otr
    @func = yield @a
    @funv = yield @b
    @s = 0
    (@n - 1).times do
      @s += yield @x
      @x += 1
    end
    @summa = @s
  end

  def trap
    ((@func + @funv) / 2 + @summa) * (@b - @a) / @n
  end
end

```

2. Программа для взаимодействия с пользователем через консоль

```ruby
# frozen_string_literal: true

require_relative 'functions'
include Math

puts 'Введите нижнюю границу'
down = gets.to_i

puts 'Введите верхнюю границу'
up = gets.to_i

a = Calculator.new(down, up)
a.calc_otr { |x| x + cos(x) }
puts "Result: #{a.trap}"

a = Calculator.new(down, up)
lam = lambda { |x|
  include Math
  x + cos(x)
}
a.calc_otr(&lam)
puts "Result: #{a.trap}"

```

3. Тест

```ruby
# frozen_string_literal: true

# test for Part 3
require_relative 'functions'

RSpec.describe Calculator do
  include Math

  it 'should return right values with -1 and 4' do
    a = Calculator.new(-1, 4)
    lam = ->(x) { x + cos(x) }
    a.calc_otr(&lam)
    expect(a.trap.round(2)).to eq(9.92)
  end

  it 'should return right values with 2 and 6' do
    b = Calculator.new(2, 6)
    lam = ->(x) { x + cos(x) }
    b.calc_otr(&lam)
    expect(b.trap.round(2)).to eq(16.38)
  end

  it 'should return right values with 1 and 2' do
    c = Calculator.new(1, 2)
    lam = ->(x) { tan(x + 1) / (x + 1) }
    c.calc_otr(&lam)
    expect(c.trap.round(2)).to eq(-0.57)
  end
end

```

4. Результат

1.1 Основной программы

```
Введите нижнюю границу
-4
Введите верхнюю границу
1
Result: 6.923848694359205
Result: 6.923848694359205
```

1.2 Теста

```
...

Finished in 0.01148 seconds (files took 0.29098 seconds to load)
3 examples, 0 failures
```

5. Проверка Rubocop:

```
Inspecting 3 files
...

3 files inspected, no offenses detected
```

6. Проверка Reek:

```
rk/IPL-Laboratory6$ reek Part\ 3
Inspecting 3 file(s):
...

0 total warnings
```

Вывод:

Я продолжил знакомиться с языком Ruby и попрактиковался в написании программ и тестов