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

Дисциплина: Языки интернет-программирования

Вариант: 23

Студент ИУ6-33Б (Группа) _____ (Подпись, дата) Д. И. Мироненко (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) В. Д. Шульман (И.О. Фамилия)

Москва, 2023

Часть 1

Задание

Решить задачу, организовав итерационный цикл с точностью $\xi = 10^{-4}, 10^{-5}$.
Вычислить сумму ряда:
$S = 1 + \sum_{k-1}^{\infty} \frac{(-1)^k}{k!}$, точное значение равно $\frac{1}{e}$.

Код программы

Файл “main.rb”

```ruby
# frozen_string_literal: true

require_relative 'utils'

eps = 1e-4
puts "#{solve(eps)} (eps = #{eps})"

eps = 1e-5
puts "#{solve(eps)} (eps = #{eps})"
```

Файл “utils.rb”

```ruby
# frozen_string_literal: true

def _factorial(num)
  return 1 if num.zero?

  result = 1
  (1..num).each do |i|
    result *= i
  end

  result
end

def solve(eps)
  expected = 1 / Math::E

  sum = 1
  k = 0
  loop do
    sum += ((-1)**(k + 1)).to_f / _factorial(k + 1)
    k += 1
    break if (expected - sum).abs <= eps.to_f
  end

  sum
end
```

Файл “test_utils.rb”

```ruby
# frozen_string_literal: true

require 'minitest/autorun'
require_relative 'utils'

# Test '_factorial'
class TestFactorial < Minitest::Test
  def test_regular
    assert_equal 1, _factorial(0)
    assert_equal 1, _factorial(1)
    assert_equal 120, _factorial(5)
  end
end

# Test 'solve'
class TestSolve < Minitest::Test
  def test_regular
    expected = 1 / Math::E

    eps = 1e-4
    assert_in_delta expected, solve(eps), eps

    eps = 1e-5
    assert_in_delta expected, solve(eps), eps
  end
end
```

Работа программы

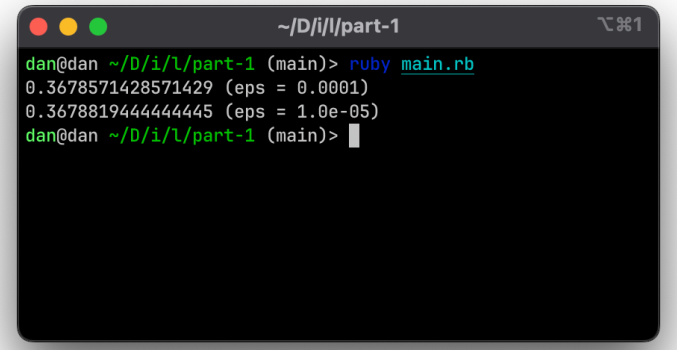

Рисунок 1 – работа программы

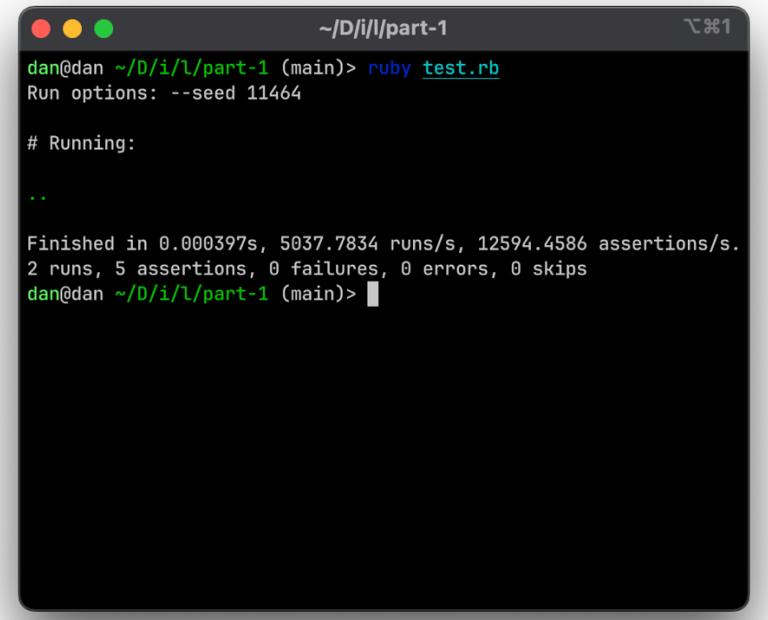

Рисунок 2 – выполнение unit-тестов

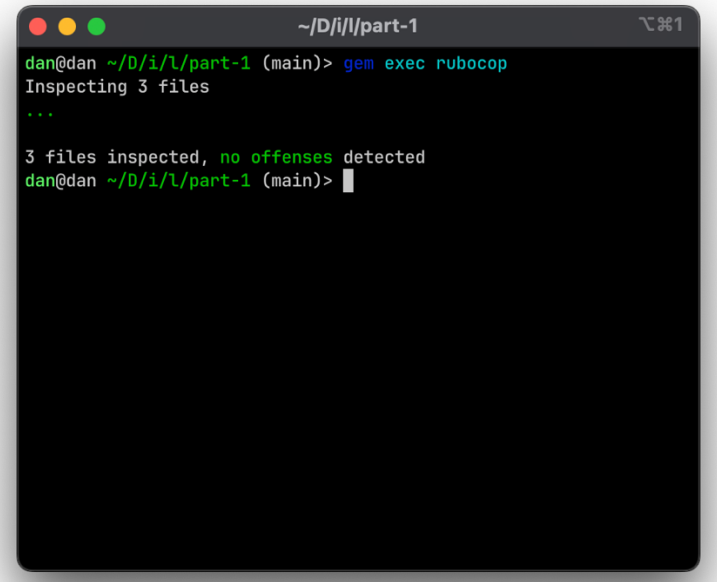

Рисунок 3 – проверка кода rubocop’ом

Часть 2

Задание

Решить предыдущее задание с помощью Enumerable или Enumerator.

Код программы

Файл “main.rb”

```
# frozen_string_literal: true

require_relative 'utils'

eps = 1e-4
puts "#{solve(eps)} (eps = #{eps})"

eps = 1e-5
puts "#{solve(eps)} (eps = #{eps})"
```

Файл “utils.rb”

```
# frozen_string_literal: true

def _factorial(num)
  return 1 if num.zero?

  (1..num).inject(:*)
end

def solve(eps)
  expected = 1 / Math::E

  sequence = Enumerator.new do |x|
    sum = 1
    k = 0
    loop do
      sum += ((-1)**(k + 1)).to_f / _factorial(k + 1)
      x << sum
      k += 1
    end
  end

  sequence.find { |item| (expected - item).abs <= eps }
end
```

Файл “test_utils.rb”

```
# frozen_string_literal: true

require 'minitest/autorun'
require_relative 'utils'

# Test '_factorial'
class TestFactorial < Minitest::Test
  def test_regular
    assert_equal 1, _factorial(0)
    assert_equal 1, _factorial(1)
    assert_equal 120, _factorial(5)
  end
end
```

```ruby
# Test 'solve'
class TestSolve < Minitest::Test
  def test_regular
    expected = 1 / Math::E

    eps = 1e-4
    assert_in_delta expected, solve(eps), eps

    eps = 1e-5
    assert_in_delta expected, solve(eps), eps
  end
end
```

Работа программы

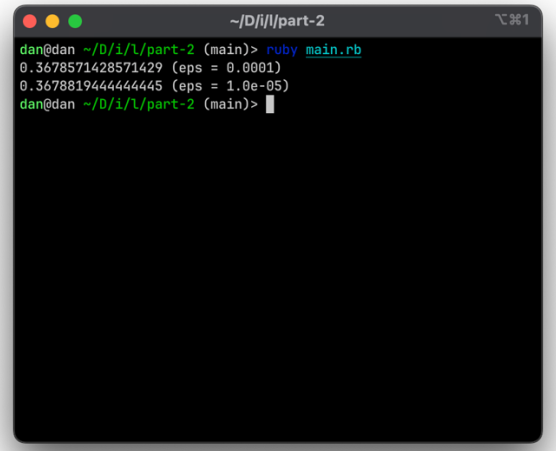

Рисунок 4 – работа программы

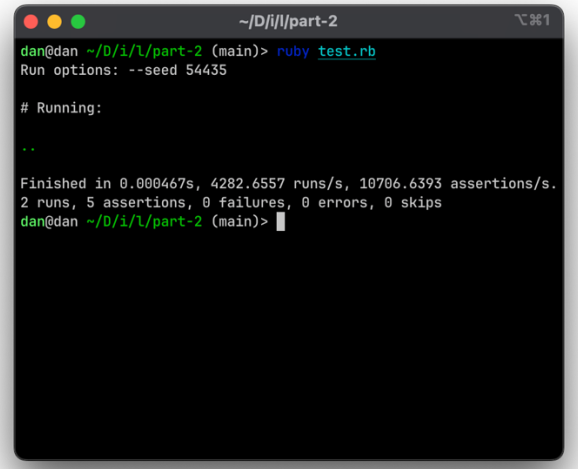

Рисунок 5 – выполнение unit-тестов

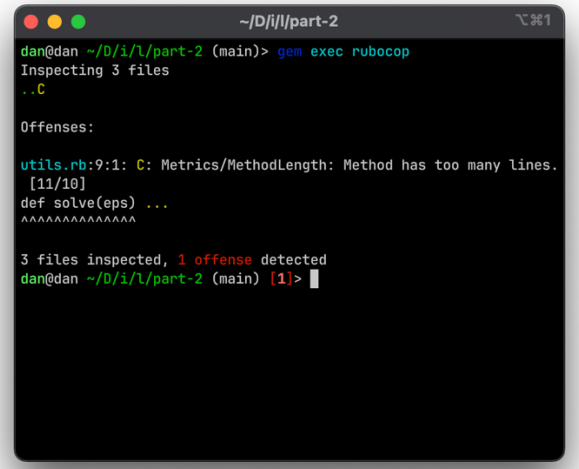

Рисунок 6 – проверка кода rubocop'ом

Часть 3

Задание

Составить метод neibr проверки принадлежности точки плоскости с координатами (x, y) данной кривой $y = f(x)$. В основной программе использовать метод neibr для проверки принадлежности десяти различных точек кривым $y = \cos(x)$ и $y = \sin(x^2)$.

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

Код программы

Файл "main.rb"

```
# frozen_string_literal: true

require_relative 'utils'

points = [
  [0, 1], [4, -0.76], [2, -0.42], [3, 0.41], [4, -0.65],
  [5, 0.28], [4, -0.28], [6, -0.99], [5, -0.13], [9, -0.91]
]

lambda_f1 = ->(x) { Math.cos(x) }
lambda_f2 = ->(x) { Math.sin(x**2) }

puts 'Lambdas:'

points.each do |x, y|
  result = neibr(x, y, &lambda_f1)
  puts "(#{x}, #{y}) belongs to y = cos(x)? - #{result}"
```

```
end

puts '-' * 45

points.each do |x, y|
  result = neibr(x, y, &lambda_f2)
  puts "(#{x}, #{y}) belongs to y = sin(x^2)? - #{result}"
end

puts "\n\nBlocks:"

points.each do |x, y|
  result = neibr(x, y) { |value| block_f1(value) }
  puts "(#{x}, #{y}) belongs to y = cos(x)? - #{result}"
end

puts '-' * 45

points.each do |x, y|
  result = neibr(x, y) { |value| block_f2(value) }
  puts "(#{x}, #{y}) belongs to y = sin(x^2)? - #{result}"
end
```

Файл “utils.rb”

```
# frozen_string_literal: true

EPS = 1e-2

def block_f1(val)
  Math.cos(val)
end

def block_f2(val)
  Math.sin(val**2)
end

def neibr(x_val, y_val, &func)
  (y_val - func.call(x_val)).abs <= EPS
end
```

Файл “test_utils.rb”

```
# frozen_string_literal: true

require 'minitest/autorun'
require_relative 'utils'

# Test 'neibr'
class TestNeibr < Minitest::Test
  def test_f1_with_lambdas
    points = [
      [0, 1], [4, -0.76], [2, -0.42], [3, 0.41], [4, -0.65],
      [5, 0.28], [4, -0.28], [6, -0.99], [5, -0.13], [9, -0.91]
    ]

    lambda_f1 = ->(x) { Math.cos(x) }
```

```
  expected = [
    true, false, true, false, true,
    true, false, false, false, true
  ]

  current_result = []
  points.each do |x, y|
    current_result << neibr(x, y, &lambda_f1)
  end

  assert_equal expected, current_result
end

def test_f2_with_lambdas
  points = [
    [0, 1], [4, -0.76], [2, -0.42], [3, 0.41], [4, -0.65],
    [5, 0.28], [4, -0.28], [6, -0.99], [5, -0.13], [9, -0.91]
  ]

  lambda_f2 = ->(x) { Math.sin(x**2) }

  expected = [
    false, false, false, true, false,
    false, true, true, true, false
  ]

  current_result = []
  points.each do |x, y|
    current_result << neibr(x, y, &lambda_f2)
  end

  assert_equal expected, current_result
end

def test_f1_with_blocks
  points = [
    [0, 1], [4, -0.76], [2, -0.42], [3, 0.41], [4, -0.65],
    [5, 0.28], [4, -0.28], [6, -0.99], [5, -0.13], [9, -0.91]
  ]

  expected = [
    true, false, true, false, true,
    true, false, false, false, true
  ]

  current_result = []
  points.each do |x, y|
    current_result << neibr(x, y) { |value| block_f1(value) }
  end

  assert_equal expected, current_result
end

def test_f2_with_blocks
  points = [
    [0, 1], [4, -0.76], [2, -0.42], [3, 0.41], [4, -0.65],
    [5, 0.28], [4, -0.28], [6, -0.99], [5, -0.13], [9, -0.91]
```

```
    ]

    expected = [
      false, false, false, true, false,
      false, true, true, true, false
    ]

    current_result = []
    points.each do |x, y|
      current_result << neibr(x, y) { |value| block_f2(value) }
    end

    assert_equal expected, current_result
  end
end
```

Работа программы

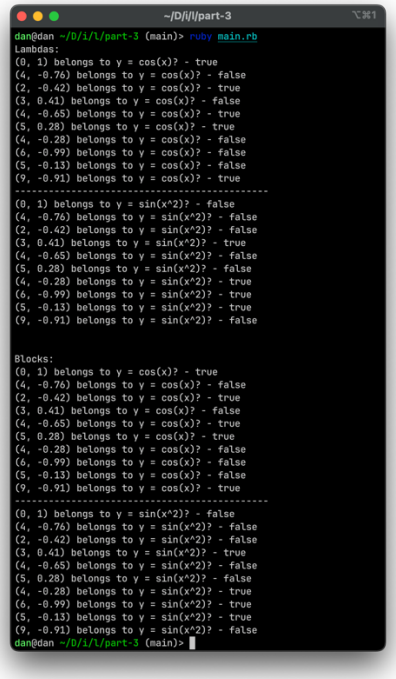

Рисунок 7 – работа программы

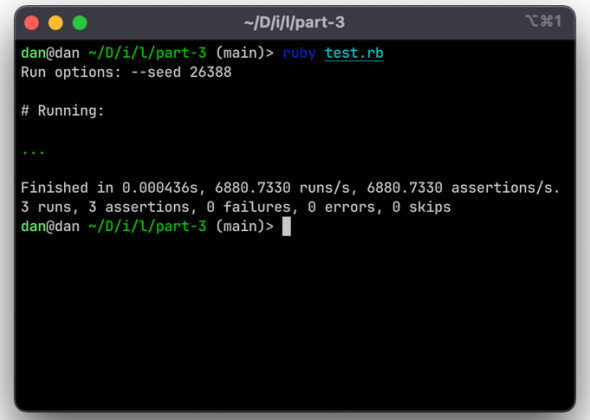

Рисунок 8 – выполнение unit-тестов

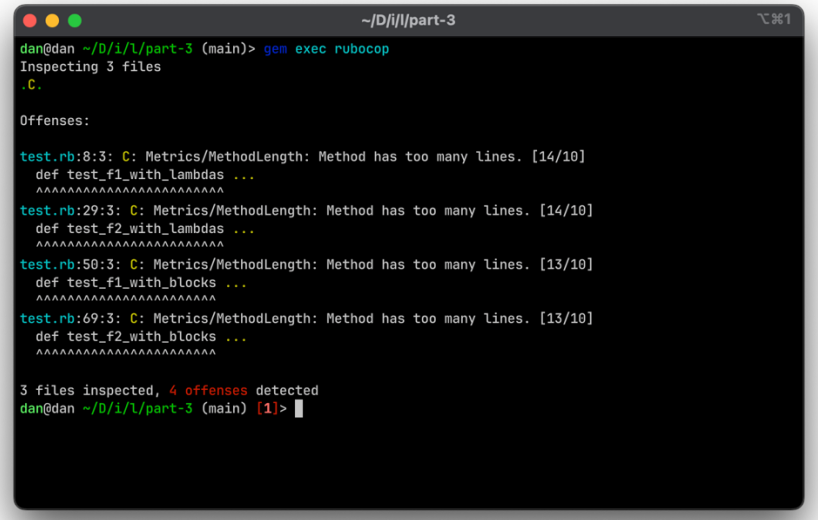

Рисунок 9 – проверка кода rubocop’ом

Вывод

Реализованы несколько программ на языке программирования ruby; функции всех программ проверена на unit-тестах; исходный код проверен линтером rubocop на наличие ошибок.