|  |  |
| --- | --- |
| 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: **09.03.01 Информатика и Вычислительная техника**

**Отчет**

**по лабораторной работе 6**

**Дисциплина: языки интернет-программирования**

**Вариант №15**

Студент гр. **ИУ6-33Б \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** В.К. Залыгин

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** В.Д. Шульман

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

**Цель**

Изучить расширенные возможности языка Ruby (инструменты потоковой обработки, примеси из стандартной библиотеки).

**Задание**

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

Каждое задание сдавать в виде:

* отдельного архива, содержащего указанные файлы;
* pdf-файл c отчётом.

Отчет должен содержать:

1. заполненный титульный лист
2. текст задания
3. тексты программ с подписанными именами файлов
4. результаты выполнения
5. результаты проверки анализаторами rubocop и reek

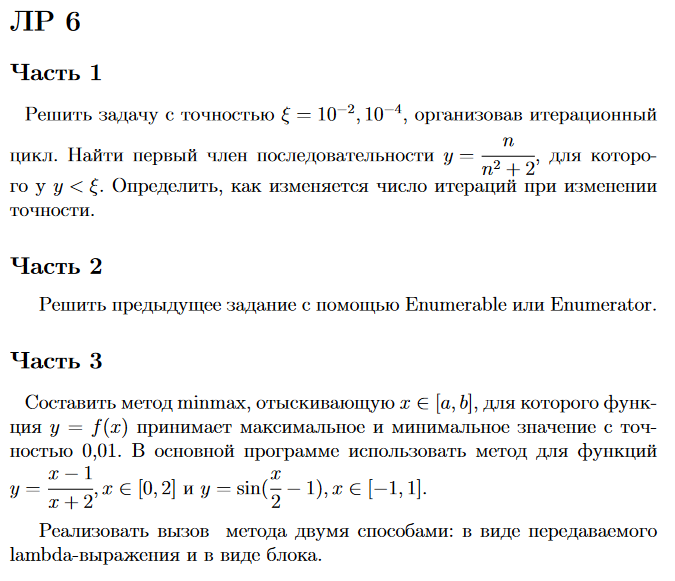


Рисунок 1 - задание для 15 варианта

**Выполнение**

Main.rb

# frozen\_string\_literal: true

require\_relative 'task1'

require\_relative 'task2'

require\_relative 'task3'

def nav\_to\_task

  puts 'select task number'

  case gets.to\_i

  when 1

    io\_task1

  when 2

    io\_task2

  when 3

    io\_task3

  else

    puts 'unknown task number'

  end

end

nav\_to\_task

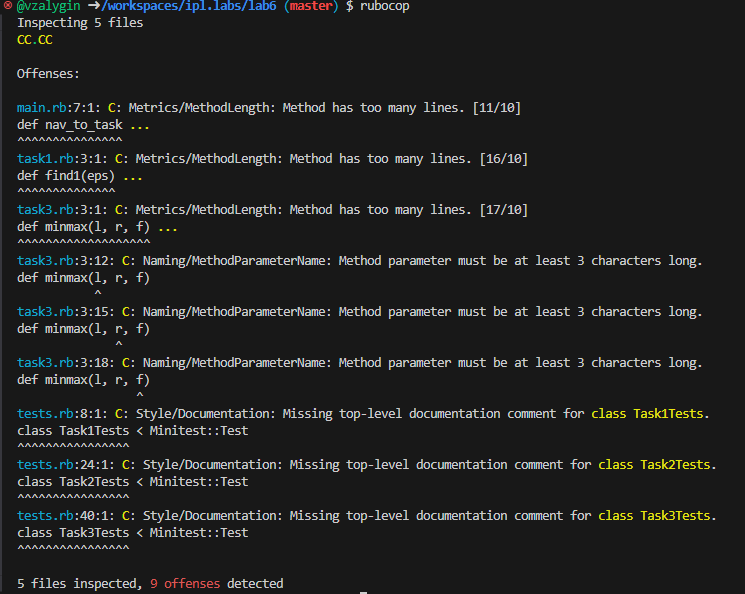


Рисунок 2 - ран рубокопа

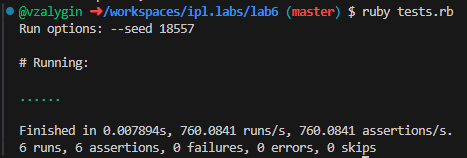


Рисунок 3 - ран тестов

**Часть 1**

Task1.rb

# frozen\_string\_literal: true

def find1(eps)

  func = ->(n) { 1.0 \* n / ((n\*\*2) + 2) }

  left = 0

  right = 1

  while func.call(right) >= eps

    left = right

    right \*= 2

  end

  while right - left > 1

    cur = (right + left) / 2

    if func.call(cur) < eps

      right = cur

    else

      left = cur

    end

  end

  right

end

def io\_task1

  puts 'enter the eps'

  eps = gets.to\_f

  puts "the first element smaller then #{eps} is on #{find1(eps)} place"

end

Тестирование:

# frozen\_string\_literal: true

require 'minitest/autorun'

require\_relative 'task1'

require\_relative 'task2'

require\_relative 'task3'

class Task1Tests < Minitest::Test

  def test\_big\_eps

    eps = 0.01

    expected = 100

    assert\_equal(expected, find1(eps))

  end

  def test\_small\_eps

    eps = 0.0001

    expected = 10\_000

    assert\_equal(expected, find1(eps))

  end

end

**Часть 2**

Task2.rb

# frozen\_string\_literal: true

def find2(eps)

  sequence = Enumerator.new do |y|

    n = 1

    loop do

      y << [1.0 \* n / ((n\*\*2) + 2), n]

      n += 1

    end

  end

  sequence.take\_while { |value\_key| value\_key[0] >= eps }[-1][1] + 1

end

def io\_task2

  puts 'enter the eps'

  eps = gets.to\_f

  puts "the first element smaller then #{eps} is on #{find2(eps)} place"

end

Тестирование:

class Task2Tests < Minitest::Test

  def test\_big\_eps

    eps = 0.01

    expected = 100

    assert\_equal(expected, find2(eps))

  end

  def test\_small\_eps

    eps = 0.0001

    expected = 10\_000

    assert\_equal(expected, find2(eps))

  end

end

**Часть 3**

# frozen\_string\_literal: true

def minmax(l, r, f)

  eps = 0.01

  max\_value = -Float::INFINITY

  max\_i = 0

  min\_value = Float::INFINITY

  min\_i = 0

  (0..((r - l) / eps).to\_i).each do |i|

    i = i \* eps + l

    if max\_value < f.call(i)

      max\_value = f.call(i)

      max\_i = i

    end

    if min\_value > f.call(i)

      min\_value = f.call(i)

      min\_i = i

    end

  end

  [min\_i, max\_i]

end

def io\_task3

  fs = [[0, 2, ->(x) { (x - 1) / (x + 2) }],

        [-1, 1, proc { |x| Math.sin(x / 2 - 1) }]]

  puts 'choose f1 (print 1) or f2 (print 2)'

  min, max = minmax(\*fs[gets.to\_i - 1])

  puts "min: #{min}, max: #{max}"

end

Тестирование:

class Task3Tests < Minitest::Test

  def test\_real

    f = [0, 2, ->(x) { (x - 1) / (x + 2) }]

    expected = [0, 2]

    assert\_equal(expected, minmax(\*f))

  end

  def test\_sin

    f = [-1, 1, proc { |x| Math.sin(x / 2 - 1) }]

    expected = [-1, 1]

    assert\_equal(expected, minmax(\*f))

  end

end

**Вывод**

Изучены расширенные возможности языка Ruby (инструменты потоковой обработки, примеси из стандартной библиотеки).