|  |  |
| --- | --- |
| 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 5**

**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 команды языка Ruby, научиться создавать простые программы с основными конструкциями передачи управления. Ознакомиться с фреймворком для написания unit-тестов.

**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

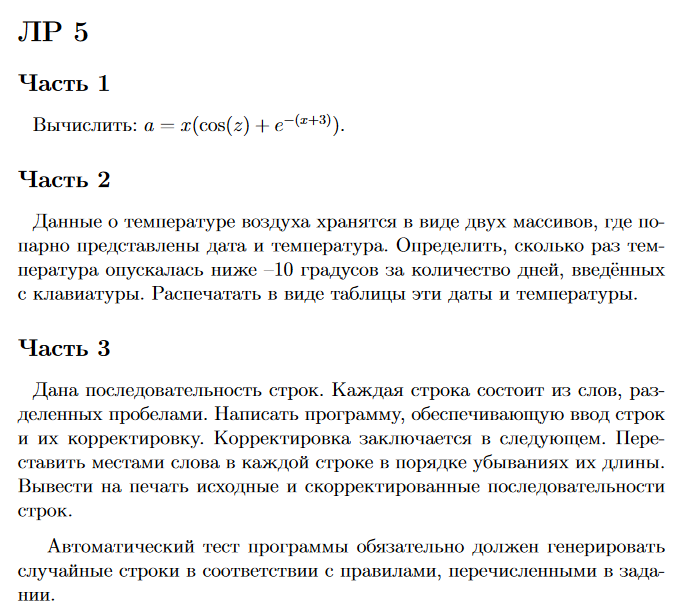


Рисунок 1 - задание для 15 варианта

**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

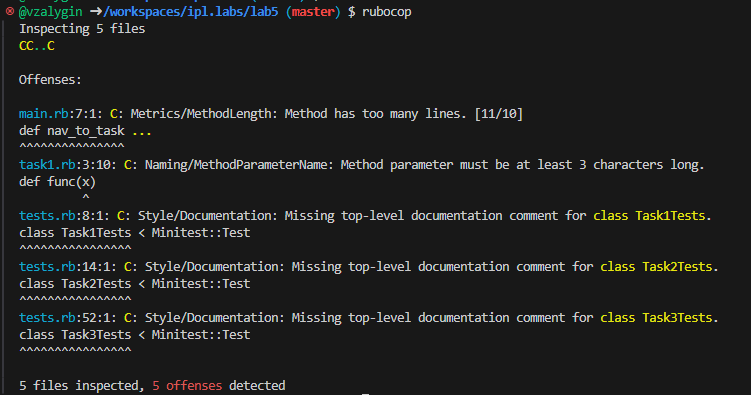


Рисунок 2 - вывод rubocop по лр

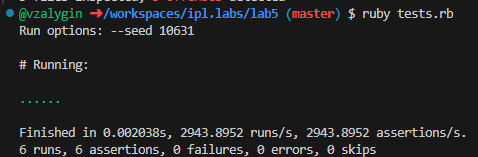


Рисунок 3 - результат тест-рана

**Часть 1**

Task1.rb

# frozen\_string\_literal: true

def func(x)

  x \* (Math.cos(x) + Math.exp(-(x + 3)))

end

def io\_task1

  puts 'enter x'

  puts "the result is: #{func(gets.to\_i)}"

end

Тестирование:

# frozen\_string\_literal: true

require 'minitest/autorun'

require\_relative 'task1'

require\_relative 'task2'

require\_relative 'task3'

class Task1Tests < Minitest::Test

  def test\_task

    assert\_equal(0, func(0))

  end

end

**Часть 2**

Task2.rb

# frozen\_string\_literal: true

def filter(dates, temps, days\_count)

  dates

    .zip(temps)

    .take(days\_count)

    .filter { |date\_temp| date\_temp[1] < 10 }

end

def input\_data(len)

  dates = []

  temps = []

  (1..len).map do |\_|

    data = gets.split

    dates.push data[0]

    temps.push data[1].to\_i

  end

  [dates, temps]

end

def io\_task2

  puts 'enter a length of data'

  len = gets.to\_i

  dates, temps = input\_data(len)

  puts 'enter a number'

  n = gets.to\_i

  puts "date\t\ttemp"

  filter(dates, temps, n)

    .each { |x| puts "#{x[0]}\t#{x[1]}" }

end

Тестирование:

class Task2Tests < Minitest::Test

  def test\_empty

    dates = []

    temps = []

    n = 0

    expected = []

    assert\_equal(expected, filter(dates, temps, n))

  end

  def test\_regular

    dates = ['01.01.1970', '02.01.1970', '03.01.1970']

    temps = [15, -5, 10]

    n = 3

    expected = [['02.01.1970', -5]]

    assert\_equal(expected, filter(dates, temps, n))

  end

  def test\_regular2

    dates = ['05.01.1970', '06.01.1970', '07.01.1970']

    temps = [15, 0, 10]

    n = 2

    expected = [['06.01.1970', 0]]

    assert\_equal(expected, filter(dates, temps, n))

  end

  def test\_regular3

    dates = ['01.01.1970', '02.01.1970', '03.01.1970']

    temps = [10, -5, -10]

    n = 1

    expected = []

    assert\_equal(expected, filter(dates, temps, n))

  end

end

**Часть 3**

Task3.rb

# frozen\_string\_literal: true

def correct\_str(str)

  str.split(' ').sort\_by { |word| -word.size }.join(' ')

end

def correct\_strs(strs)

  strs.map { |str| correct\_str(str) }

end

def io\_task3

  puts 'input number of strings'

  n = gets.to\_i

  strs = (1..n).map { |\_| gets }

  puts "before modification:\n"

  strs.each { |str| puts str }

  strs = correct\_strs(strs)

  puts "after modification:\n"

  strs.each { |str| puts str }

end

Тестирование:

class Task3Tests < Minitest::Test

  def test\_empty

    strs = []

    expected = []

    assert\_equal(expected, correct\_strs(strs))

  end

  def regular\_test

    strs = ['abc ab a', 'a ac abc', 'a a a', 'a a ab']

    expected = ['abc ab a', 'abc ac a', 'a a a', 'ab a a']

    assert\_equal(expected, correct\_strs(strs))

  end

  def generate\_random\_word

    (1..rand(1, 15)).map { |\_| ('a'.ord + rand(26)).chr }.join('')

  end

  def generate\_random\_str

    (1..rand(1, 15)).map { |\_| generate\_random\_word }.join(' ')

  end

end

**Вывод**

Изучены команды языка Ruby, получен навык написания простых программ с основными конструкциями передачи управления. Проведено ознакомление с фреймворком для написания unit-тестов.