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 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 с 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

ЛР 5

Часть 1

Вычислить: $a = x(\cos(z) + e^{-(x+3)})$.

Часть 2

Данные о температуре воздуха хранятся в виде двух массивов, где попарно представлены дата и температура. Определить, сколько раз температура опускалась ниже –10 градусов за количество дней, введённых с клавиатуры. Распечатать в виде таблицы эти даты и температуры.

Часть 3

Дана последовательность строк. Каждая строка состоит из слов, разделенных пробелами. Написать программу, обеспечивающую ввод строк и их корректировку. Корректировка заключается в следующем. Переставить местами слова в каждой строке в порядке убываниях их длины. Вывести на печать исходные и скорректированные последовательности строк.

Автоматический тест программы обязательно должен генерировать случайные строки в соответствии с правилами, перечисленными в задании.

Рисунок 1 - задание для 15 варианта

Выполнение

Main.rb

```ruby
# 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
```

```
@vzalygin ➜/workspaces/ipl.labs/lab5 (master) $ rubocop
Inspecting 5 files
CC..C

Offenses:

main.rb:7:1: C: Metrics/MethodLength: Method has too many lines. [11/10]
def nav_to_task ...
^^^^^^^^^^^^^^^
task1.rb:3:10: C: Naming/MethodParameterName: Method parameter must be at least 3 characters long.
def func(x)
         ^
tests.rb:8:1: C: Style/Documentation: Missing top-level documentation comment for class Task1Tests.
class Task1Tests < Minitest::Test
^^^^^^^^^^^^^^^^
tests.rb:14:1: C: Style/Documentation: Missing top-level documentation comment for class Task2Tests.
class Task2Tests < Minitest::Test
^^^^^^^^^^^^^^^^
tests.rb:52:1: C: Style/Documentation: Missing top-level documentation comment for class Task3Tests.
class Task3Tests < Minitest::Test
^^^^^^^^^^^^^^^^

5 files inspected, 5 offenses detected
```

Рисунок 2 - вывод rubocop по лр

```
@vzalygin ➜/workspaces/ipl.labs/lab5 (master) $ ruby tests.rb
Run options: --seed 10631

# Running:

......

Finished in 0.002038s, 2943.8952 runs/s, 2943.8952 assertions/s.
6 runs, 6 assertions, 0 failures, 0 errors, 0 skips
```

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'
```

```ruby
  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:

```ruby
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.