|  |  |
| --- | --- |
| **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 **09.03.03 «Прикладная информатика»**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 8 |

**Название:**

Создание каркаса RoR-приложения. Настройка

базовых параметров. Размещение файлов.

                       Создание форм. Функциональные тесты.

**Дисциплина:** Языки Интернет-программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-34Б |  | 15.10.2022 | С. А. Рахманов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Д. В. Малахов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

**Цель работы:** получение навыков программирования на языке Ruby с использованием функционального стиля программирования, ознакомиться со средствами проверки соответствия стиля программирования.

**Задание:**

Все консольные приложения Ruby следует реализовывать в виде трех

отдельных файлов:

1. основная программа;

2. программа для взаимодействия с пользователем через консоль;

3. программа для автоматического тестирования на основе MiniTest::Unit

или RSpec. Везде, где это возможно, данные для проверки должны

формироваться автоматически по правилам, указанным в задании.

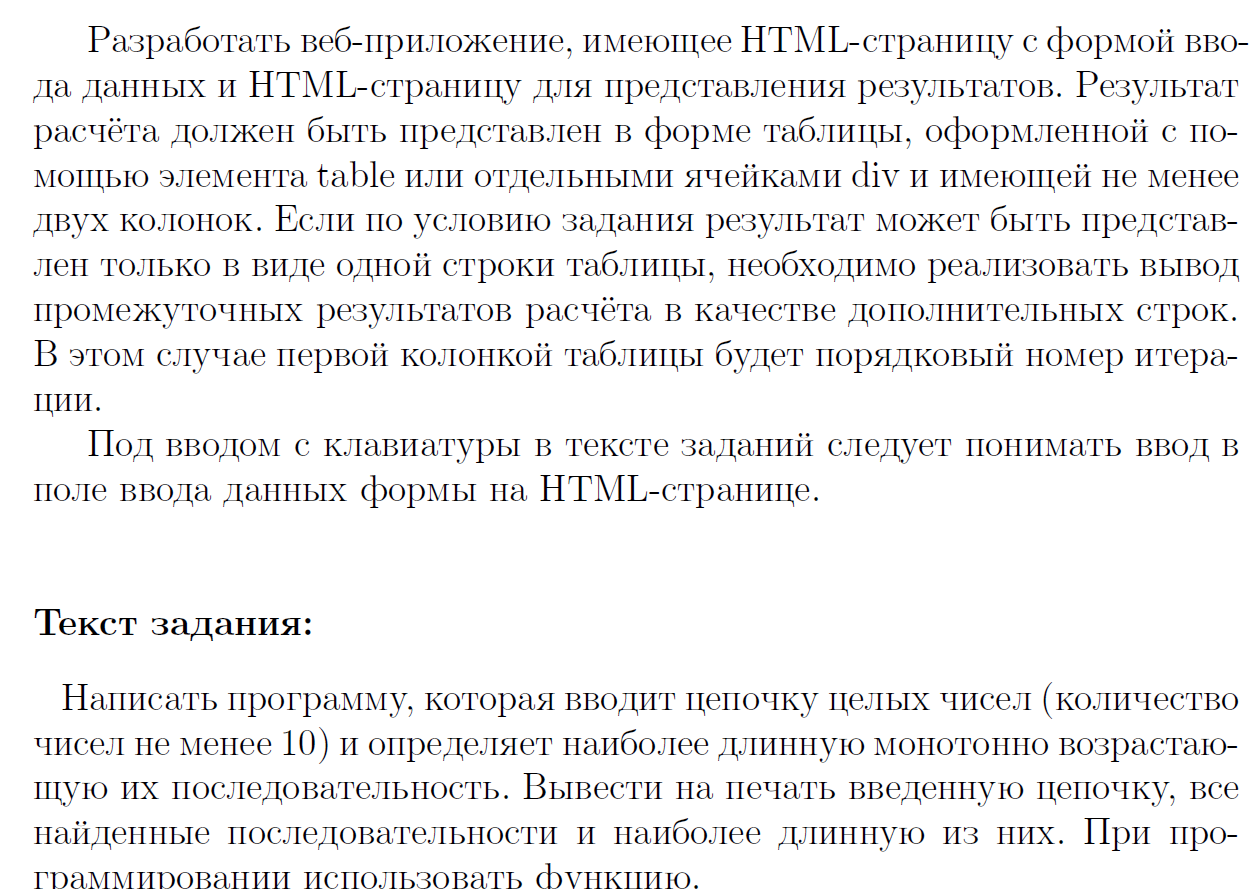
При реализации программ везде, где это возможно, следует избегать

использования циклов for, do, while. Вместо них используйте методы из

примеси Enumerable.

Все тексты программ должны быть проверены на соответствие стилю

программирования Ruby при помощи rubocop.ru или reek.



**Тексты измененных файлов**

*seq\_controller.rb*

# frozen\_string\_literal: true

def correct (arr)

found = []

tmp\_found = []

arr.take(arr.size - 1).each\_index do |i|

if arr[i] < arr [i+1]

tmp\_found << arr[i]

else

tmp\_found << arr[i]

found << tmp\_found

tmp\_found = []

end

end

tmp\_found << arr[-1]

found << tmp\_found

sizes = found.map { |x| x.size }

@max\_found = found[sizes.index(sizes.max)].join(' ')

@found = found.map { |x| x.join(' ') }.join(' | ')

end

class SeqController < ApplicationController

def input

end

def show

@str = params[:str]

@max\_found, @found = @str, @str

arr = @str&.split&.map(&:to\_i)

arr && not(arr&.empty?) ? correct(arr) : nil

end

end

*input.html.erb*

<h1>Seq#input</h1>

<p>Find me in app/views/seq/input.html.erb</p>

<%= form\_tag("/seq/show", :method => "get") do %>

<%= label\_tag("Input string:") %>

<%= text\_field\_tag(:str) %> <br/>

<br/>

<%= submit\_tag("Correct and print") %>

<% end %>

*show.html.erb*

<h1>Seq#show</h1>

<p>Find me in app/views/seq/show.html.erb</p>

<p id="str">str: <%= @str %></p>

<p id="max found: ">max found: <%= @max\_found %></p>

<p id="found: ">found: <%= @found %></p>

<%= link\_to "Repeat input", '/' %>

*routes.rb*

Rails.application.routes.draw do

root 'seq#input', as: :home

get 'seq/show'

# Define your application routes per the DSL in https://guides.rubyonrails.org/routing.html

# Defines the root path route ("/")

# root "articles#index"

end

*seq\_controller\_test.rb*

# frozen\_string\_literal: true

require "test\_helper"

Test\_data = [

{

input: {str: '1 2 3 4 3 4 5 1 1 1 5 3 4 5 6 10 0 -1 -3'},

output: ['1 2 3 4 3 4 5 1 1 1 5 3 4 5 6 10 0 -1 -3', '3 4 5 6 10', '1 2 3 4 | 3 4 5 | 1 | 1 | 1 5 | 3 4 5 6 10 | 0 | -1 | -3']

},

{

input: {str: '1 2 3 4 1 2 3 1 2'},

output: ['1 2 3 4 1 2 3 1 2', '1 2 3 4', '1 2 3 4 | 1 2 3 | 1 2']

},

{

input: {str: '38 23 20 28 49 22 39 9 3 56 4'},

output: ['38 23 20 28 49 22 39 9 3 56 4', '20 28 49', '38 | 23 | 20 28 49 | 22 39 | 9 | 3 56 | 4']

}

]

class SeqControllerTest < ActionDispatch::IntegrationTest

test "should get show" do

get '/seq/show'

assert\_response :success

end

test "should get input" do

get '/'

assert\_response :success

end

test "should return empty result" do

input = {str: ''}

get '/seq/show?' + input.to\_query

assert\_equal('', assigns(:str))

assert\_equal('', assigns(:max\_found))

assert\_equal('', assigns(:found))

end

test "should return correct data" do

Test\_data.each do |data|

input = data[:input]

output = data[:output]

get '/seq/show?' + input.to\_query

assert\_equal(output[0], assigns(:str))

assert\_equal(output[1], assigns(:max\_found))

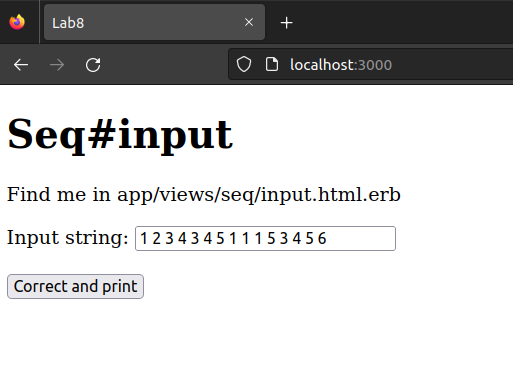
assert\_equal(output[2], assigns(:found))

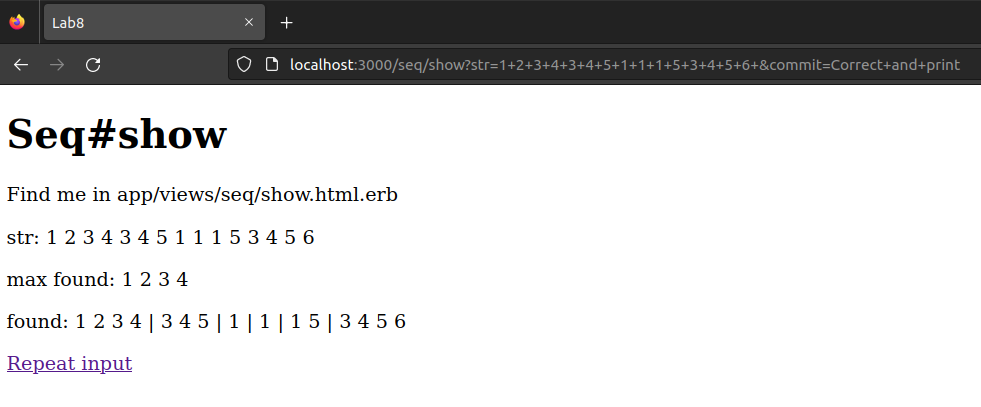
end

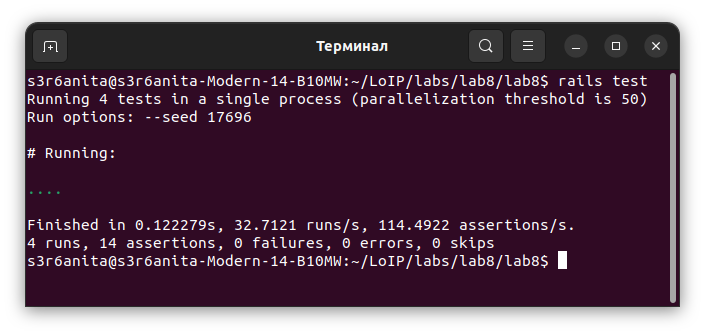
end

end

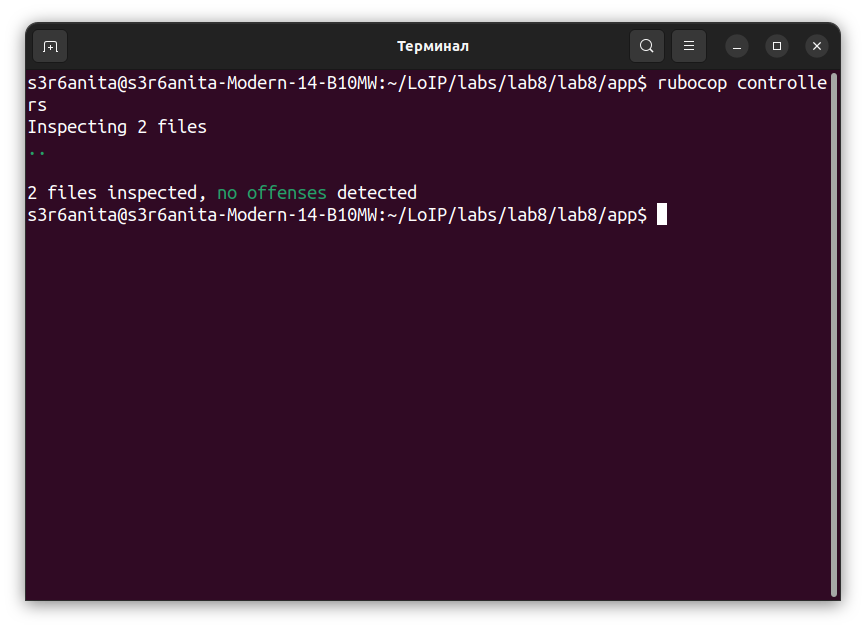
**Результаты выполнения**



****

****

**Результаты проверки анализатором rubocop:**

****

**Вывод:** получил навыки программирования на языке Ruby с использованием функционального стиля программирования, получил навыки использования rubocop и проверил написанный код на соответствие стилю программирования на Ruby.