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**

О Т Ч Е Т

по лабораторной работе № 5

Название: Работа с массивами и строками в Ruby

Дисциплина: Языки Интернет-программирования

Студент	ИУ6-35Б		29.09.2023	В. И. Мамыкин
	(Группа)		(Подпись, дата)	(И.О. Фамилия)
Преподаватель				Е.Ю. Гаврилова
			(Подпись, дата)	(И.О. Фамилия)

Москва, 2023

Задание:

Часть 1

Вычислить: $a = 1 + |y - x| + \frac{(y-x)^2}{2} + \frac{|y-x|^3}{3}$.

Часть 2

Ведомость на зарплату представлена как два массива. Один содержит фамилии работников цеха, а второй – их зарплату за текущий месяц. Найдите фамилию работника, зарплата которого наименее отклоняется от средней зарплаты всех работников за текущий месяц. Найдите фамилии двух работников с наибольшей зарплатой. Удалите из ведомости на зарплату сведения о работнике, зарплата которого минимальна.

Часть 3

Дана последовательность строк. Каждая строка состоит из слов, разделенных пробелами, в конце — точка. Слова в строке образуют пары: каждое первое слово — заменяемое, каждое второе — замещающее. Написать программу, обеспечивающую ввод строк и их корректировку. Корректировка заключается в замене всех заменяемых слов замещающими. Вывести на печать исходную и скорректированную последовательности строк.

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

Цель: реализовать консольные приложения на Ruby и написать для них тесты. Научиться использовать базовые модули Ruby, работать с массивами и строками.

Часть 1:

1.rb

```
# frozen_string_literal: true


require_relative 'functions'
```

```
puts 'Введите x: '
x = gets.chop.to_f
puts 'Введите y: '
y = gets.chop.to_f
print "Результат: #{format('%0.5f', my_function(x, y))}"
```

functions.rb

```
def my_function(x, y)
  1 + (y - x).abs + (y - x)**2 / 2 + (y - x).abs**3 / 3
End
```

tests.rb

```
# frozen_string_literal: true

require 'minitest/autorun'
require_relative '1'
class TestFunc < Minitest::Test
  def test_func1
    assert_in_delta 2.83333, my_function(1.0, 2.0), 0.001
  end

  def test_func2
    assert_in_delta 4.75000, my_function(1.0, 2.5), 0.001
  end

  def test_func3
    assert_in_delta 2.83333, my_function(2.0, 1.0), 0.001
  end
end
```

Часть 2:

1.rb

```
# frozen_string_literal: true

require_relative 'functions'
surnames = Array.new(5)
puts 'Введите фамилии: '
surnames.each_index { |i| surnames[i] = gets.chop }
paycheks = Array.new(5)
summ = 0
puts 'Введите зарплаты: '
paycheks.each_index do |i|
  paycheks[i] = gets.chop.to_f
  summ += paycheks[i]
end
sr_arifm = summ / 5
puts 'Фамилия работника, зарплата котрого наименее отклоняется от средней:
'
puts operation_1(surnames, paycheks, sr_arifm)
puts 'Фамилии работников, зарплаты которых наибольшие: '
puts operation_2(surnames, paycheks)
puts 'Фамилии работников (исключен работник с наименьшей зарплатой): '
print operation_3(surnames, paycheks)
```

functions.rb

```
# frozen_string_literal: true

def operation_1(surnames, paychecks, sr_arifm)
  deviation = 1_000_000
  index = 0
```

```
  paychecks.each_index do |i|
    if (paychecks[i] - sr_arifm).abs < deviation
      deviation = (paychecks[i] - sr_arifm).abs
      index = i
    end
  end
  surnames[index]
end

def operation_2(surnames, paychecks)
  maximum = 0
  pred_maximum = 0
  paychecks.each_index do |i|
    if paychecks[i] > paychecks[maximum]
      pred_maximum = maximum
      maximum = i
    elsif paychecks[i] > paychecks[pred_maximum]
      pred_maximum = i
    end
  end
  [surnames[pred_maximum], surnames[maximum]]
end

def operation_3(surnames, paycheks)
  minimum = 0
  paycheks.each_index do |i|
    minimum = i if paycheks[i] < paycheks[minimum]
  end
  surnames.delete_at(minimum)
  surnames
```

```
end
```

tests.rb

```
# frozen_string_literal: true



require 'minitest/autorun'

require_relative '1'

class TestFunc < Minitest::Test

  def test_func1

    assert_equal('c', operation_1(%w[a b c d e], [1, 2, 3, 4, 5], 3))

    assert_equal(%w[d e], operation_2(%w[a b c d e], [1, 2, 3, 4, 5]))

    assert_equal(%w[b c d e], operation_3(%w[a b c d e], [1, 2, 3, 4, 5]))

  end

end
```

Часть 3:

1.rb

```
# frozen_string_literal: true

require_relative 'functions'
my_string = gets.chop
```

```
my_array = my_string.split
puts "Исходная строка: #{my_string}"
my_array[-1] = my_array[-1].delete_suffix('.')
# print my_array
# puts
puts "Полученная строка: #{permutations(my_array)}"
# print my_array
# s = ''
# 0.upto(rand(1..10) * 2) { s += (1..rand(30)).map { rand(97..122).chr }.join + ' '
# s = s.delete_suffix(' ')
# s += '.'
# puts s
```

functions.rb

```
# frozen_string_literal: true

def permutations(my_arr)
  my_array = my_arr.clone
  my_array.each_index do |i|
    my_array[i] = my_array[i + 1] unless i.odd?
  end
  my_array.join(' ')
end

def proverka(my_arr)
  flag = true
  my_array = my_arr.clone
  my_array2 = my_arr.clone
  my_array2.each_index do |i|
    my_array2[i] = my_array2[i + 1] unless i.odd?
  end
```

```
  my_array2.each_index do |i|
    flag = false if !i.odd? && (my_array2[i] != my_array[i - 1])
  end
  if flag
    my_array.join(' ')
  else
    my_array2.join(' ')
  end
end
```

tests.rb

```
# frozen_string_literal: true

require 'minitest/autorun'
require_relative '1'
class TestFunc < Minitest::Test
  def test_func1
    s = ''
    1.upto(rand(1..10) * 2) { s += (1..rand(30)).map { rand(97..122).chr }.join + ' ' }
    s = s.delete_suffix(' ')
    s += '.'
    my_arr = s.split
    my_arr2 = s.split
    my_arr[-1] = my_arr[-1].delete_suffix('.')
    my_arr2[-1] = my_arr2[-1].delete_suffix('.')
    print 'test: ' + s
    assert_equal(proverka(my_arr), permutations(my_arr2))
  end
end
```

```
C:\Ruby32-x64\bin\ruby.exe D:/Documents/GitHub/BMSTU/3semestr/YAIP/lab5/1/1.rb
Введите x:
10
Введите y:
20
Результат: 394.33333
Process finished with exit code 0
```

Рисунок 1 – Результат выполнения 1 функции

```
C:\Ruby32-x64\bin\ruby.exe D:/Documents/GitHub/BMSTU/3semestr/YAIP/lab5/2/1.rb
Введите фамилии:
A
B
C
D
E
Введите зарплаты:
1
2
3
4
5
Фамилия работника, зарплата котрого наименее отклоняется от средней:
C
Фамилии работников, зарплаты которых наибольшие:
D
E
Фамилии работников (исключен работник с наименьшей зарплатой):
["B", "C", "D", "E"]
Process finished with exit code 0
```

Рисунок 2 – Результат выполнения 2 функции

```
C:\Ruby32-x64\bin\ruby.exe D:/Documents/GitHub/BMSTU/3semestr/YAIP/lab5/3/1.rb
a b c d
Исходная строка: a b c d
Полученная строка: b b d d

Process finished with exit code 0
```

Рисунок 3 – Результат выполнения 3 функции

```
# Running:

...

Finished in 0.012301s, 243.8806 runs/s, 243.8806 assertions/s.
3 runs, 3 assertions, 0 failures, 0 errors, 0 skips
```

Рисунок 4 – Результат работы тестов 1 части

```
# Running:

.

Finished in 0.012819s, 78.0086 runs/s, 234.0258 assertions/s.
1 runs, 3 assertions, 0 failures, 0 errors, 0 skips
```

Рисунок 5 – Результат работы тестов 2 части

```
# Running:

test: dkoxfpafmldxcbkjscbxgrmsuucwz askrstr k br glvrhokkmlqbql mirmrr k
aatzs mrupltphevpvb aogroichzwjaayzffoyqiyive vqhyqqtcpdmedexldjg hzivcv
yvtwsonvuu cjcrturogxo kffxizp lmcemgwewcmbol..

Finished in 0.013328s, 75.0311 runs/s, 75.0311 assertions/s.
1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
```

Рисунок 6 – Результат работы тестов 3 части (с отладочной печатью)

```
Inspecting 3 files

...

3 files inspected, no offenses detected
```

Рисунок 7 – Результат работы robocop *–config robocop.yaml (для всех частей задания)*

```
Inspecting 3 file(s):

...

0 total warnings
```

Рисунок 8 – Результат работы reek (для всех частей задания)

Вывод: были сделаны консольные приложения на Ruby и написаны для них тесты. Изучено использование базовых модули Ruby, а также работа с массивами и строками.