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 № 6

Название: Использование методов, Enumerable, Enumerator в Ruby

Дисциплина: Языки Интернет-программирования

Студент	ИУ6-35Б		13.10.2023	В. И. Мамыкин
	(Группа)		(Подпись, дата)	(И.О. Фамилия)
Преподаватель				Е.Ю. Гаврилова
			(Подпись, дата)	(И.О. Фамилия)

Москва, 2023

Задание:

Часть 1

Решить задачу с точностью $\xi = 10^{-3}, 10^{-4}$, организовав итерационный цикл. Вычислить значение определенного интеграла методом прямоугольников: $\int_1^2 ln\ x\ \mathrm{d}x$. Считать точным значением: 0,3862943611199. Определить, как изменяется число итераций при изменении точности.

Часть 2

Решить предыдущее задание с помощью Enumerable или Enumerator.

Часть 3

Составить метод intprg вычисления определенного интеграла по формуле прямоугольников: $S = \frac{b-a}{n}\sum_{i=1}^{n} f(x_i)$, где n – количество отрезков разбиения. В основной программе использовать метод intprg для вычисления интегралов: $\int_0^1 \frac{e^x}{x+1}\,\mathrm{d}x$ и $\int_0^2 x(x-1)\,\mathrm{d}x$.

Реализовать вызов метода двумя способами: в виде передаваемого lambda-выражения и в виде блока.

Цель: реализовать консольные приложения на Ruby и написать для них тесты. Научиться использовать Enumerable и Enumerator, lambda функции и блоки.

Часть 1:

1.rb

```
# frozen_string_literal: true


require_relative 'functions'
a = 1
b = 2
puts "Значение с точностью eps = 0.001: #{integral(a, b, 0.001)}"
```

```
puts "Значение с точностью eps = 0.0001: #{integral(a, b, 0.0001)}"
```

functions.rb

```
# frozen_string_literal: true

$number_of_iteration = 0
def F(x)
  Math.log(x)
end

def get_value(a, b, n)
  h = ((b - a) * 1.0 / n)
  res = 0
  i = 1
  n.times do
    x = a + h * i
    res += F(x)
    i += 1
    $number_of_iteration += 1
  end
  res *= h
  res
end

# def integral(a, b, eps)
#   n = 2
#   cur = get_value(a, b, n)
#   prev = -10_000
#   while (cur - prev).abs > eps
#     $number_of_iteration = 0
#     prev = cur
```

```
#     n *= 2
#     cur = get_value(a, b, n)
#   end
#   puts "Количество итераций : #{$number_of_iteration}"
#   cur
# end
#
def integral(a, b, eps, n = 2, prev = -10_000)
  cur = get_value(a, b, n)
  if (cur - prev).abs <= eps
    puts "Количество итераций: #{$number_of_iteration}"
    cur
  else
    $number_of_iteration = 0
    integral(a, b, eps, n * 2, cur)
  end
end
```

test.rb

```
# frozen_string_literal: true

require 'minitest/autorun'
require_relative '1'
class TestFunc < Minitest::Test
  def test_func1
    assert_in_delta 0.3862943611199, integral(1, 2, 0.001), 0.001
    assert_in_delta 0.3862943611199, integral(1, 2, 0.0001), 0.0001
  end
end
```

Часть 2:

1.rb

```
# frozen_string_literal: true

require_relative 'functions'
a = 1
b = 2
puts "Значение с точностью eps = 0.001: #{integral(a, b, 0.001)}"
puts "Значение с точностью eps = 0.0001: #{integral(a, b, 0.0001)}"
```

functions.rb

```
# frozen_string_literal: true

$number_of_iteration = 0
def F(x)
  Math.log(x)
end

def get_value(a, b, n)
  h = ((b - a) * 1.0 / n)
  res = 0
  1.upto(n) do |i|
    x = a + h * i
    res += F(x)
    $number_of_iteration += 1
  end
  res *= h
  res
end

def integral(a, b, eps)
  cur = get_value(a, b, 2)
```

```
  prev = -10_000
  (2..20).map { |i| 2**i }.each do |i|
    $number_of_iteration = 0
    prev = cur
    cur = get_value(a, b, i)
    break if (cur - prev).abs < eps
  end
  puts "Количество итераций : #{$number_of_iteration}"
  cur
end
```

test.rb

```
# frozen_string_literal: true

require 'minitest/autorun'
require_relative '1'
class TestFunc < Minitest::Test
  def test_func1
    assert_in_delta 0.3862943611199, integral(1, 2, 0.001), 0.001
    assert_in_delta 0.3862943611199, integral(1, 2, 0.0001), 0.0001
  end
end
```

Часть 3:

1.rb

```
# frozen_string_literal: true

require_relative 'functions'
n = 2048
puts "Вычисление через блок: #{intprg(0, 1) do |a, b|
  res = 0
```

```
  h = (b - a) * 1.0 / n
  1.upto(n) do |i|
    x = a + h * i
    res += Math::E**x / (x + 1)
  end
  res *= h
end}"
value = lambda do |a, b|
  res = 0
  h = (b - a) * 1.0 / n
  1.upto(n) do |i|
    x = a + h * i
    res += x * (x - 1)
  end
  res *= h
end
puts "Вычисление через lambda-функцию: #{intprg(0, 2, &value)}"
```

functions.rb

```
# frozen_string_literal: true
def intprg(a, b)
    yield a, b
  end
```

test.rb

```
# frozen_string_literal: true

require 'minitest/autorun'
require_relative '1'
class TestFunc < Minitest::Test
  def test_func1
    n = 2048
```

```
    value1 = intprg 0, 1 do |a, b|
      res = 0
      h = (b - a) * 1.0 / n
      1.upto(n) do |i|
        x = a + h * i
        res += Math::E**x / (x + 1)
      end
      res *= h
    end
    value2 = lambda do |a, b|
      res = 0
      h = (b - a) * 1.0 / n
      1.upto(n) do |i|
        x = a + h * i
        res += x * (x - 1)
      end
      res *= h
    end
    assert_in_delta 1.1256, value1, 0.001
    assert_in_delta 2 * 1.0 / 3, intprg(0, 2, &value2), 0.001
  end
end
```

```
C:\Ruby32-x64\bin\ruby.exe D:/Documents/GitHub/BMSTU/3semestr/YAIP/lab6/1/1.rb
Количество итераций: 512
Значение с точностью eps = 0.001: 0.38697110371772203
Количество итераций: 4096
Значение с точностью eps = 0.0001: 0.3863789713293034

Process finished with exit code 0
```

Рисунок 1 – Результат выполнения первой функции

```
C:\Ruby32-x64\bin\ruby.exe D:/Documents/GitHub/BMSTU/3semestr/YAIP/lab6/2/1.rb
Количество итераций : 512
Значение с точностью eps = 0.001: 0.38697110371772203
Количество итераций : 4096
Значение с точностью eps = 0.0001: 0.3863789713293034

Process finished with exit code 0
```

Рисунок 2 – Результат выполнения 2 функции

```
C:\Ruby32-x64\bin\ruby.exe D:/Documents/GitHub/BMSTU/3semestr/YAIP/lab6/3/1.rb
Вычисление через блок: 1.1254737774723846
Вычисление через lambda-функцию: 0.6676435470581055

Process finished with exit code 0
```

Рисунок 3 – Результат выполнения 3 функции

```
# Running:

Количество итераций: 512
Количество итераций: 4096
.

Finished in 0.014897s, 67.1276 runs/s, 134.2552 assertions/s.
1 runs, 2 assertions, 0 failures, 0 errors, 0 skips
```

Рисунок 4 – Результат выполнения тестов 1 части

```
# Running:

Количество итераций : 512
Количество итераций : 4096
.

Finished in 0.015071s, 66.3522 runs/s, 132.7043 assertions/s.
1 runs, 2 assertions, 0 failures, 0 errors, 0 skips
```

Рисунок 5 – Результат выполнения тестов 2 части

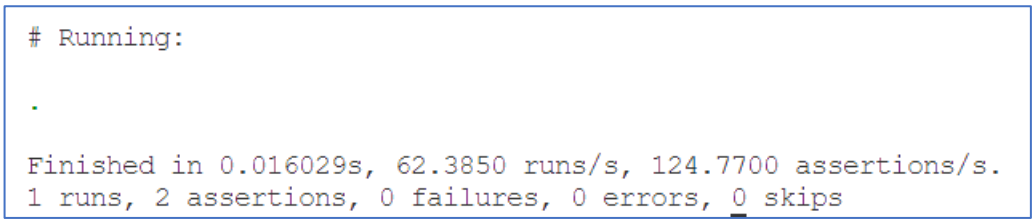

Рисунок 6 – Результат выполнения тестов 3 части

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

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

Вывод: были сделаны консольные приложения на Ruby и написаны для них тесты. Изучены принципы работы с Enumerable и Enumerator, lambda функциями и блоками.