|  |  |
| --- | --- |
| 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 №** | **6** |

**Название:**

Использование методов, Enumerable, Enumerator в Ruby

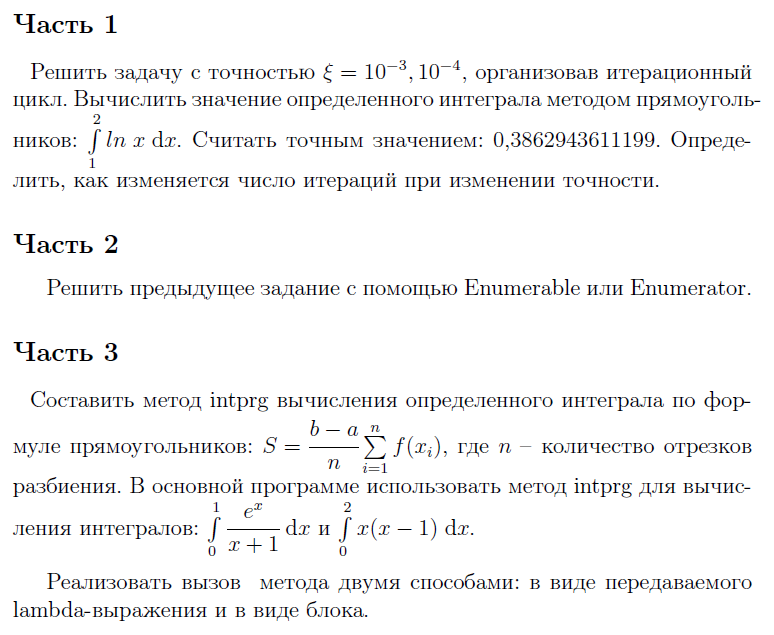
**Дисциплина:** Языки Интернет-программирования



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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:**

****

**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

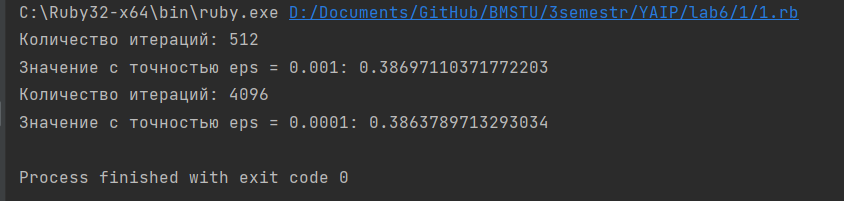


Рисунок 1 – Результат выполнения первой функции

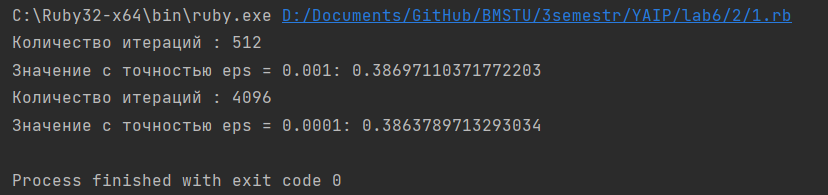


Рисунок 2 – Результат выполнения 2 функции

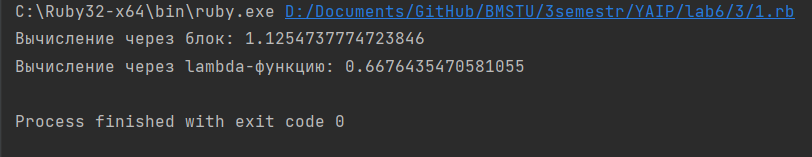


Рисунок 3 – Результат выполнения 3 функции

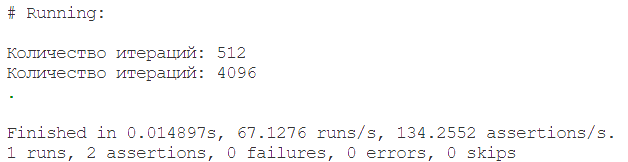


Рисунок 4 – Результат выполнения тестов 1 части

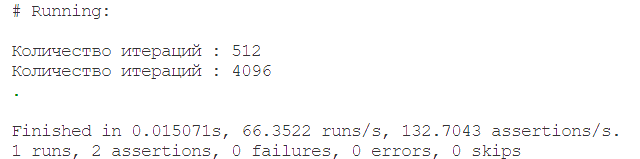


Рисунок 5 – Результат выполнения тестов 2 части

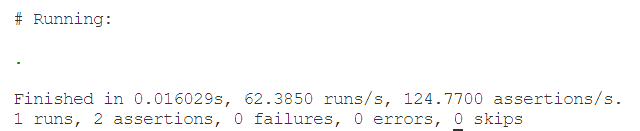


Рисунок 6 – Результат выполнения тестов 3 части

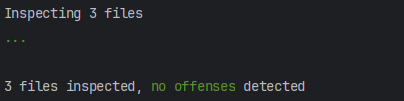


Рисунок 7 – Результат работы robocop (*rubocop --config rubocop.yaml*) (аналогично для всех частей задания)



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.