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**

ОТЧЕТ

по лабораторной работе № 6

Название: Ruby - Enumerable, Enumerator, блоки

Дисциплина: Языки Интернет-программирования

Студент ИУ-32б _____ С.В. Астахов
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ _____
(Подпись, дата) (И.О. Фамилия)

Москва, 2020

Задание 1

Решить задачу, организовав итерационный цикл с точностью $\xi = 10^{-3}, 10^{-4}$. Вычислить площадь круга как предел последовательности площадей правильных вписанных многоугольников с удваивающимся числом сторон. Формула для нахождения площади правильного n-угольника: $S_n = \frac{1}{2}R^2 n \sin\frac{2\pi}{n}$. Определить, как изменяется число итераций при изменении точности.

Исходный код
Count.rb (логика)

```
#!/usr/bin/ruby
# frozen_string_literal: true

# counts given function
module Counter
  def self.count(radius, epsilon)
        sides = 2
        square = 0
        while ((radius**2) * Math::PI - square) > epsilon
          sides += 1
          square = 0.5 * (radius**2) * sides * Math.sin(2 * Math::PI / sides)
        end
        sides - 2
  end
end
```

Src.rb (консольный интерфейс)

```
# frozen_string_literal: true

require_relative 'count'
require_relative 'count'

puts("\nApproximation of circle square \n\n")
10.times do |variable|
  printf("eps: %<eps>1.10f iter-s: %<res>d \n\n", eps: 10**-(variable + 1), res:
Counter.count(1, 10**-(variable + 1)))
end
```

Test.rb (автотесты)

```
# frozen_string_literal: true

require './count.rb'
require 'minitest/autorun'

# inherits default test class
class TestFunc < Minitest::Test
  def setup; end

  def test_dependence
        assert Counter.count(1, 0.01) < Counter.count(1, 0.001), 'Iterations != f(eps)'
  end

  def test_zero
        assert_equal Counter.count(0, 0.01), 0, 'Zero radius not => 0'
  end

  def test_brute
        assert_equal Counter.count(2, 12), 1, 'Not one itertion for big eps'
  end

  def test_accurate
        assert Counter.count(3, 0.00001) > 1000, 'Not over 1k itertion for low eps'
  end

  def teardown; end
end
```

Результат выполнения с произвольными входными данными

```
Approximation of circle square

eps: 0.1000000000 iter-s: 13

eps: 0.0100000000 iter-s: 44

eps: 0.0010000000 iter-s: 142

eps: 0.0001000000 iter-s: 453

eps: 0.0000100000 iter-s: 1436
```

eps: 0.0000010000 iter-s: 4545

eps: 0.0000001000 iter-s: 14376

eps: 0.0000000100 iter-s: 45464

eps: 0.0000000010 iter-s: 143772

eps: 0.0000000001 iter-s: 454651

Результаты автотестов

Run options: --seed 31216

Running:

....

Finished in 0.005373s, 744.4631 runs/s, 744.4631 assertions/s.
4 runs, 4 assertions, 0 failures, 0 errors, 0 skips

Вывод Rubocop

Inspecting 3 files
...

3 files inspected, no offenses detected

Вывод Reek

count.rb -- 1 warning:
  [9, 11]:DuplicateMethodCall: Counter#self.count calls 'radius**2' 2 times [https://github.com/troessner/reek/blob/v6.0.1/docs/Duplicate-Method-Call.md]
1 total warning

Задание 2

Решить предыдущее задание с помощью Enumerable или Enumerator.

Исходный код
Count.rb (логика)

```
#!/usr/bin/ruby
# frozen_string_literal: true

# counts given function
module Counter
  def self.count_enum(radius)
        Enumerator.new do |result|
        n_sides = 2
        loop do
        result << 0.5 * (radius**2) * n_sides * Math.sin(2 * Math::PI / n_sides)
        n_sides += 1
        end
        end
  end

  def self.count(radius, epsilon)
        count_enum(radius).take_while { |square| ((radius**2) * Math::PI -
square).abs > epsilon }.length
  end
end
```

Src.rb (консольный интерфейс)

```
# frozen_string_literal: true

require_relative 'count'

# include directive for module to use functions as built-in

my_enumerator = Enumerator.new do |table|
  eps = 0.1
  loop do
        table << [eps, Counter.count(1, eps)]
        eps /= 10
```

```
  end
end

my_enumerator.take(10).each do |elem|
  printf("eps: %<eps>1.10f iter-s: %<res>d \n\n",
         eps: elem[0],
         res: elem[1])
end
```

Test.rb (автотесты)

```
# frozen_string_literal: true

require './count.rb'
require 'minitest/autorun'

# inherits default test class
class TestFunc < Minitest::Test
  def setup; end

  def test_dependence
         assert Counter.count(1, 0.01) < Counter.count(1, 0.001), 'Iterations != f(eps)'
  end

  def test_zero
         assert_equal Counter.count(0, 0.01), 0, 'Zero radius not => 0'
  end

  def test_brute
         assert_equal Counter.count(2, 12), 1, 'Not one itertion for big eps'
  end

  def test_accurate
         assert Counter.count(3, 0.00001) > 1000, 'Not over 1k itertion for low eps'
  end

  def test_algorythm_equality
         assert_equal Counter.count(1, 0.1), 13, 'Not same accuracy as standart cycle'
  end

  def teardown; end
end
```

Результат выполнения с произвольными входными данными

eps: 0.1000000000 iter-s: 13

eps: 0.0100000000 iter-s: 44

eps: 0.0010000000 iter-s: 142

eps: 0.0001000000 iter-s: 453

eps: 0.0000100000 iter-s: 1436

eps: 0.0000010000 iter-s: 4545

eps: 0.0000001000 iter-s: 14376

eps: 0.0000000100 iter-s: 45464

eps: 0.0000000010 iter-s: 143772

eps: 0.0000000001 iter-s: 454651

Результаты автотестов

Run options: --seed 29953

Running:

.....

Finished in 0.006445s, 775.8313 runs/s, 775.8313 assertions/s.
5 runs, 5 assertions, 0 failures, 0 errors, 0 skips

Вывод Rubocop

Inspecting 3 files
...

3 files inspected, no offenses detected

Вывод Reek

Inspecting 3 file(s):
...

0 total warnings

Задание 3

Составить метод maxim для определения максимального расстояния между двумя кривыми $F(x)$ и $G(x)$ в точке $x \in [a, b]$. В основной программе использовать метод maxim для функций $\frac{\sin(x)}{x}$ и $\frac{tg(x+1)}{x+1}$ на интервале $[0.5 \ldots 1]$ с шагом 0.01.

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

Исходный код
Maxim.rb (логика)

```
#!/usr/bin/ruby
# frozen_string_literal: true

# counts max distance between functions
module Distance
  def self.maxim(function_lambda, start, stop, step_size, &function_block)
        max_distance = 0
        (start..stop).step(step_size) do |arg|
        max_distance = (function_lambda.call(arg) - function_block.call(arg)).abs \
        if (function_lambda.call(arg) - function_block.call(arg)).abs > max_distance
        end
        max_distance
  end
end
```

Src.rb (консольный интерфейс)

```
# frozen_string_literal: true

require_relative 'maxim'

func1 = ->(x) { Math.sin(x) / x }
printf("\n\nMax distance between sin(x)/x and tg(x+1)/(x+1) is:\n %<res>3.2f\n\n",\
        res: Distance.maxim(func1, 0.5, 1, 0.01) { |x| Math.tan(x + 1) / (x + 1) })
```

Test.rb (автотесты)

```
# frozen_string_literal: true

require './maxim.rb'
require 'minitest/autorun'

# inherits default test class
class TestFunc < Minitest::Test
  def setup
        @function_lambda = ->(arg) { arg * 2 }
  end

  def test_zero_distance
        assert_in_delta(Distance.maxim(@function_lambda, 0.5, 1, 0.01) { |num| num *
2 }, 0, 0.01)
  end

  def test_counts
        assert_in_delta(Distance.maxim(@function_lambda, 0.5, 1, 0.01) { |num| num *
3 }, 1, 0.01)
  end

  def test_lamda_block
        assert_in_delta(Distance.maxim(@function_lambda, 0.5, 1, 0.01) { |num| num *
4 }, 2, 0.01)
  end

  def test_infinity
        assert Distance.maxim(@function_lambda, 0.5, 1, 0.01) { |num| 1 / (num - 1)} ==
Float::INFINITY
  end

  def test_2_lamdas
        function_lambda2 = ->(arg) { arg * 3 }
        assert_in_delta(Distance.maxim(@function_lambda, 0.5, 1, 0.01,
&function_lambda2), 1, 0.01)
  end

  def test_default_functions
        function_lambda = ->(arg) { Math.sin(arg) / arg}
        assert_in_delta(Distance.maxim(function_lambda, 0.5, 1, 0.01) { |num|
Math.tan(num + 1) / (num + 1)}, 798.90, 0.01)
  end

  def teardown; end
end
```

Результат выполнения с произвольными входными данными

Max distance between sin(x)/x and tg(x+1)/(x+1) in [0.5 .. 1] is:
 798.90

Результаты автотестов

Run options: --seed 44299

Running:

.....

Finished in 0.004187s, 1432.8700 runs/s, 1432.8700 assertions/s.
6 runs, 6 assertions, 0 failures, 0 errors, 0 skips

Вывод Rubocop

Inspecting 3 files
...

3 files inspected, no offenses detected

Вывод Reek

maxim.rb -- 5 warnings:
  [9, 10]:DuplicateMethodCall: Distance#self.maxim calls '(function_lambda.call(arg) - function_block.call(arg)).abs' 2 times [https://github.com/troessner/reek/blob/v6.0.1/docs/Duplicate-Method-Call.md]
  [9, 10]:DuplicateMethodCall: Distance#self.maxim calls 'function_block.call(arg)' 2 times [https://github.com/troessner/reek/blob/v6.0.1/docs/Duplicate-Method-Call.md]
  [9, 10]:DuplicateMethodCall: Distance#self.maxim calls 'function_lambda.call(arg) - function_block.call(arg)' 2 times [https://github.com/troessner/reek/blob/v6.0.1/docs/Duplicate-Method-Call.md]
  [9, 10]:DuplicateMethodCall: Distance#self.maxim calls 'function_lambda.call(arg)' 2 times [https://github.com/troessner/reek/blob/v6.0.1/docs/Duplicate-Method-Call.md]
  [6]:LongParameterList: Distance#self.maxim has 4 parameters [https://github.com/troessner/reek/blob/v6.0.1/docs/Long-Parameter-List.md]

Вывод

В ходе данной работы были изучены основны работы с классом Enumrable, передача функций и блоков в качестве параметров.