



**«Московский государственный технический университет  
имени Н.Э. Баумана»**

**(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**

**Дисциплина: Языки Интернет-программирования**

**Название лабораторной работы: Ruby. Массивы и строковая обработка.  
Вариант 10.**

Студент гр. ИУ6-34Б

\_\_\_\_\_  
(Подпись, дата)

К.Ю. Каташинский

\_\_\_\_\_  
(И.О. Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

\_\_\_\_\_  
(И.О. Фамилия)

Москва, 2020

# Часть 1

Условие задания:

Решить задачу, организовав итерационный цикл. Вычислить длину окружности с точностью  $\xi = 10^{-3}, 10^{-4}$  как предел последовательности периметров вписанных правильных многоугольников с удваивающимся числом сторон (начать с  $n = 6$ ). Использовать формулу удвоения стороны

$$n\text{-угольника: } a_{2n} = \sqrt{2R^2 - 2R\sqrt{R^2 - a_n^2/4}}.$$

Код файла main.rb:

```
# frozen_string_literal: true
def calculate(radius, eps)
  side = radius
  i = 6 prev = sum = i * side
  loop do
    prev = sum
    side = Math.sqrt(2 * radius * radius - 2 * radius * Math.sqrt(radius * radius -
side * side * 1.0 / 4))
    i *= 2 sum = i * side
    break if (prev - sum).abs < eps
  end
  sum
end
```

Код файла test.rb:

```
# frozen_string_literal: true
require 'minitest/autorun'
require './main.rb'
# Test class
class Test < MiniTest::Unit::TestCase
  def setup
    @radius = Random.rand(200)
  end
  def test_1
    eps = 1e-3
    length = calculate(@radius, eps).round(3)
    main_length = (2 * Math::PI * @radius).round(3)
    assert_equal(length, main_length)
  end
  def test_2
    eps = 1e-4
    length = calculate(@radius, eps).round(4)
    main_length = (2 * Math::PI * @radius).round(4)
    assert_equal(length, main_length)
  end
end
```

Код файла user.rb:

```
# frozen_string_literal: true
require './main.rb'
eps = 1e-3
# eps = 1e-4
print 'Введите радиус: '
radius = gets.chomp.to_f
length = calculate(radius, eps)
print 'length = ', length.to_s
```

```
Введите радиус: 2
length = 12.566230431646488
```

Рисунок 1 «Результаты выполнения программы»

```
Inspecting 3 files
...
3 files inspected, no offenses detected
```

Рисунок 2 «Результаты проверки Rubosor»

```
Inspecting 3 file(s):
SS.

main.rb -- 5 warnings:
[9, 9]:DuplicateMethodCall: calculate calls '2 * radius' 2 times [https://github.com/troessner/reek/blob/v6.0.1/docs/Duplicate-Method-Call.md]
[6, 11]:DuplicateMethodCall: calculate calls 'i * side' 2 times [https://github.com/troessner/reek/blob/v6.0.1/docs/Duplicate-Method-Call.md]
[6, 10, 11]:FeatureEnvy: calculate refers to 'i' more than self (maybe move it to another class?) [https://github.com/troessner/reek/blob/v6.0.1/docs/Feature-Envy.md]
[3]:TooManyStatements: calculate has approx 10 statements [https://github.com/troessner/reek/blob/v6.0.1/docs/Too-Many-Statements.md]
[5, 10]:UncommunicativeVariableName: calculate has the variable name 'i' [https://github.com/troessner/reek/blob/v6.0.1/docs/Uncommunicative-Variable-Name.md]
test.rb -- 3 warnings:
[7]:InstanceVariableAssumption: Test assumes too much for instance variable '@radius' [https://github.com/troessner/reek/blob/v6.0.1/docs/Instance-Variable-Assumption.md]
[12]:UncommunicativeMethodName: Test#test_1 has the name 'test_1' [https://github.com/troessner/reek/blob/v6.0.1/docs/Uncommunicative-Method-Name.md]
[19]:UncommunicativeMethodName: Test#test_2 has the name 'test_2' [https://github.com/troessner/reek/blob/v6.0.1/docs/Uncommunicative-Method-Name.md]

8 total warnings
```

Рисунок 3 «Результаты проверки Reek»

## Часть 2

Условие задания:

*Решить предыдущее задание с помощью Enumerable или Enumerator.*

Код файла main.rb:

```
# frozen_string_literal: true
# Calculate class
class Calculate
  include Enumerable
  def initialize(radius)
    @radius = radius
  end
  def each
    side = @radius
    i = 6
    prev = sum = i * side
    loop do
      prev = sum
      side = Math.sqrt(2 * @radius * @radius - 2 * @radius * Math.sqrt(@radius *
@radius - side * side * 1.0 / 4))
      i *= 2
      sum = i * side
      yield prev, sum
    end
  end
end
```

Код файла test.rb:

```
require 'minitest/autorun'
require './main.rb'

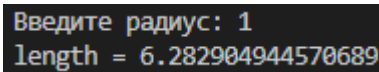
# Test class
class Test < Minitest::Unit::TestCase
  def setup
    @radius = Random.rand(200)
  end
  def test_1
    eps = 1e-3
    length = Calculate.new(@radius).find { |prev, sum| (prev - sum).abs < eps }
    length = length[1].round(3)
    main_length1 = (2 * Math::PI * @radius).round(3)
    assert_equal(length, main_length1)
  end
  def test_2
    eps = 1e-4
    length = Calculate.new(@radius).find { |prev, sum| (prev - sum).abs < eps }
    length = length[1].round(4)
    main_length1 = (2 * Math::PI * @radius).round(4)
    assert_equal(length, main_length1)
  end
end
```

Код файла user.rb:

```
# frozen_string_literal: true
require './main.rb'
eps = 1e-3
# eps = 1e-4
```

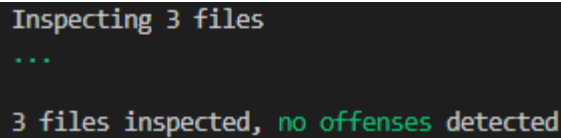
Продолжение кода файла user.rb:

```
print 'Введите радиус: '  
radius = gets.chomp.to_f  
length = Calculate.new(radius).find { |prev, sum| (prev - sum).abs < eps }  
print 'length = ', length[1]
```



```
Введите радиус: 1  
length = 6.282904944570689
```

Рисунок 4 «Результаты выполнения программы»



```
Inspecting 3 files  
...  
3 files inspected, no offenses detected
```

Рисунок 5 «Результаты проверки Rubosop»



```
Inspecting 3 file(s):  
SS.  
  
main.rb -- 4 warnings:  
[17, 17]:DuplicateMethodCall: Calculate#each calls '2 * @radius' 2 times [https://github.com/troessner/reek/blob/v6.0.1/docs/Duplicate-Method-Call.md]  
[14, 19]:DuplicateMethodCall: Calculate#each calls 'i * side' 2 times [https://github.com/troessner/reek/blob/v6.0.1/docs/Duplicate-Method-Call.md]  
[11]:TooManyStatements: Calculate#each has approx 9 statements [https://github.com/troessner/reek/blob/v6.0.1/docs/Too-Many-Statements.md]  
[13, 18]:UncommunicativeVariableName: Calculate#each has the variable name 'i' [https://github.com/troessner/reek/blob/v6.0.1/docs/Uncommunicative-Variable-Name.md]  
test.rb -- 7 warnings:  
[7]:InstanceVariableAssumption: Test assumes too much for instance variable '@radius' [https://github.com/troessner/reek/blob/v6.0.1/docs/Instance-Variable-Assumption.md]  
[12]:TooManyStatements: Test#test_1 has approx 6 statements [https://github.com/troessner/reek/blob/v6.0.1/docs/Too-Many-Statements.md]  
[20]:TooManyStatements: Test#test_2 has approx 6 statements [https://github.com/troessner/reek/blob/v6.0.1/docs/Too-Many-Statements.md]  
[12]:UncommunicativeMethodName: Test#test_1 has the name 'test_1' [https://github.com/troessner/reek/blob/v6.0.1/docs/Uncommunicative-Method-Name.md]  
[20]:UncommunicativeMethodName: Test#test_2 has the name 'test_2' [https://github.com/troessner/reek/blob/v6.0.1/docs/Uncommunicative-Method-Name.md]  
[16]:UncommunicativeVariableName: Test#test_1 has the variable name 'main_length1' [https://github.com/troessner/reek/blob/v6.0.1/docs/Uncommunicative-Variable-Name.md]  
[24]:UncommunicativeVariableName: Test#test_2 has the variable name 'main_length1' [https://github.com/troessner/reek/blob/v6.0.1/docs/Uncommunicative-Variable-Name.md]  
11 total warnings
```

Рисунок 6 «Результаты проверки Reek»

## Часть 3

Условие задания:

Составить метод `intg` 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 `intg` для вычисления интегралов:  $\int_{0,1}^1 \frac{\sin x}{x} dx$

и  $\int_1^2 \frac{\operatorname{tg}(x+1)}{x+1} dx$ .

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

Код файла `main.rb`:

```
# frozen_string_literal: true
def intg(b, a, func)
  h = 1e-5
  n = (b - a) / h
  result = 0
  x = a
  (1..n).each do |i|
    result += block_given? ? yield(x) : func.call(x)
    x += h
  end
  result * (b - a) * 1.0 / n
end
```

Код файла `test.rb`:

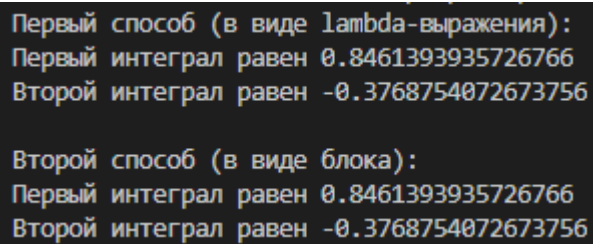
```
# frozen_string_literal: true
require 'minitest/autorun'
require './main.rb'
# Test class
class Test < MiniTest::Unit::TestCase
  def test_1
    func1 = ->(x) { Math.sin(x) * 1.0 / x }
    intg1 = intg(1, 0.1, func1) { |x| Math.sin(x) * 1.0 / x }
    assert_equal(intg(1, 0.1, func1), intg1)
  end
  def test_2
    func2 = ->(x) { Math.tan(x + 1) * 1.0 / (x + 1) }
    intg2 = intg(2, 1, func2) { |x| Math.tan(x + 1) * 1.0 / (x + 1) }
    assert_equal(intg(2, 1, func2), intg2)
  end
end
```

Код файла `user.rb`:

```
# frozen_string_literal: true
require './main.rb'
puts 'Первый способ (в виде lambda-выражения): '
func1 = ->(x) { Math.sin(x) * 1.0 / x }
func2 = ->(x) { Math.tan(x + 1) * 1.0 / (x + 1) }
puts 'Первый интеграл равен ' + intg(1, 0.1, func1).to_s
puts 'Второй интеграл равен ' + intg(2, 1, func2).to_s
puts "\nВторой способ (в виде блока): "
```

Продолжение кода файла user.rb:

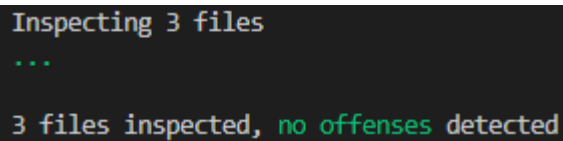
```
intg1 = intg(1, 0.1, func1) { |x| Math.sin(x) * 1.0 / x}
intg2 = intg(2, 1, func2) { |x| Math.tan(x + 1) * 1.0 / (x + 1)}
puts 'Первый интеграл равен ' + intg1.to_s
puts 'Второй интеграл равен ' + intg2.to_s
```



```
Первый способ (в виде lambda-выражения):
Первый интеграл равен 0.8461393935726766
Второй интеграл равен -0.3768754072673756

Второй способ (в виде блока):
Первый интеграл равен 0.8461393935726766
Второй интеграл равен -0.3768754072673756
```

Рисунок 7 «Результаты выполнения программы»



```
Inspecting 3 files
...

3 files inspected, no offenses detected
```

Рисунок 8 «Результаты проверки Rubosor»



```
Inspecting 3 file(s):
SS.

main.rb -- 9 warnings:
[5, 12]:DuplicateMethodCall: intg calls 'b - a' 2 times [https://github.com/troessner/reek/blob/v6.0.1/docs/Duplicate-Method-Call.md]
[5, 12]:FeatureEnvy: intg refers to 'b' more than self (maybe move it to another class?) [https://github.com/troessner/reek/blob/v6.0.1/docs/Feature-Envy.md]
[9, 12]:FeatureEnvy: intg refers to 'result' more than self (maybe move it to another class?) [https://github.com/troessner/reek/blob/v6.0.1/docs/Feature-Envy.md]
[3]:TooManyStatements: intg has approx 9 statements [https://github.com/troessner/reek/blob/v6.0.1/docs/Too-Many-Statements.md]
[3]:UncommunicativeParameterName: intg has the parameter name 'a' [https://github.com/troessner/reek/blob/v6.0.1/docs/Uncommunicative-Parameter-Name.md]
[3]:UncommunicativeParameterName: intg has the parameter name 'b' [https://github.com/troessner/reek/blob/v6.0.1/docs/Uncommunicative-Parameter-Name.md]
[4]:UncommunicativeVariableName: intg has the variable name 'h' [https://github.com/troessner/reek/blob/v6.0.1/docs/Uncommunicative-Variable-Name.md]
[5]:UncommunicativeVariableName: intg has the variable name 'n' [https://github.com/troessner/reek/blob/v6.0.1/docs/Uncommunicative-Variable-Name.md]
[7, 10]:UncommunicativeVariableName: intg has the variable name 'x' [https://github.com/troessner/reek/blob/v6.0.1/docs/Uncommunicative-Variable-Name.md]
```

Рисунок 9 «Результаты проверки Reek»

## Вывод

В ходе этой лабораторной работы изучен Enumerable и использован в решении задач, изучены способы передачи выражений в другие функции: с помощью lambda-выражений и с помощью блока в Ruby.