Федеральное государственное бюджетное образовательное учреждение высшего образования

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

(национальный исследовательский университет)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

О т ч е т

по лабораторной работе № 6

Дисциплина: Языки Интернет-программирования.

Студент гр. ИУ6-33Б _____ **Дасов Т.Д.**

(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ _____

(Подпись, дата) (И.О. Фамилия)

Москва, 2019

Часть 1

Вычислить $arctg(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$ в точке $x = 0,5$ и с точностью $\xi = 10^{-3}, 10^{-4}$. Определить, как изменяется число итераций с изменением точности.

Код основной программы:

```
# frozen_string_literal: true

# Math module
class MyMath
  class << self
    def arctg(x_val, accuracy)
      enu = Enum.new(x_val, 3, 1, x_val)
      puts enu.res
      loop do
        enu = iterate(enu, x_val)
        break if (enu.res - enu.prev).abs < accuracy
      end
      enu
    end

    private

    def iterate(enu, x_val)
      enu.prev = enu.res
      enu.res += (-1)**enu.s_val * ((x_val**enu.n_val) / enu.n_val)
      enu.s_val = enu.s_val == 1 ? 2 : 1
      enu.n_val += 2
      enu
    end
  end

  # Class doc
  class Enum
    attr_accessor :res, :n_val, :s_val, :prev

    def initialize(res, n_val, s_val, prev)
      @res = res
      @n_val = n_val
      @s_val = s_val
      @prev = prev
    end
  end
end
```

Программа для взаимодействия с пользователем:

```
# frozen_string_literal: true

require_relative 'my_math.rb'

# Input controller class
class Input
```

```
  def self.start
    puts 'Введите x:'
    x = gets.to_f
    puts 'Введите точность:'
    ac = gets.to_f
    r = MyMath.arctg(x, ac)
    puts "result: #{r.res} (accuracy: #{ac}, iterations: #{(r.n_val - 3) / 2})"
  end
end

Input.start
```

Тесты

```
# frozen_string_literal: true

require 'test/unit'
require_relative 'my_math.rb'

# Test class
class Prog1Test < Test::Unit::TestCase
  def setup; end

  def test1
    puts 'Введите количество и точность тестов:'
    n = gets.to_i
    ac = gets.to_f
    n.times do
      r = rand(-0.9..0.9)
      assert((MyMath.arctg(r, ac).res - Math.atan(r)).abs <= ac)
    end
  end
end
```

Часть 2

Решить предыдущее задание с помощью Enumerator.

Код основной программы:

```
# frozen_string_literal: true

require 'ostruct'
# MyMath documentation
class MyMath
  def each
    res = 0.5
    n = 3
    s = 1
    loop do
      res += (-1)**s * ((0.5**n) / n)
      yield res if block_given?
      s = s == 1 ? 2 : 1
      n += 2
    end
  end

  class << self
```

```
    def calc_arctg(x_val, accuracy)
      Enumerator.new do |y|
        enu = Enum.new(x_val, 3, 1, x_val)
        loop do
          enu = iterate(enu, x_val)
          y.yield enu.res, (enu.n_val - 3) / 2
          break if (enu.res - enu.prev).abs < accuracy
        end
      end
    end

    private

    def iterate(enu, x_val)
      enu.prev = enu.res
      enu.res += (-1)**enu.s_val * ((x_val**enu.n_val) / enu.n_val)
      enu.s_val = enu.s_val == 1 ? 2 : 1
      enu.n_val += 2
      enu
    end
  end
end

# Enumeration class
class Enum
  attr_accessor :res, :n_val, :s_val, :prev

  def initialize(res, n_val, s_val, prev)
    @res = res
    @n_val = n_val
    @s_val = s_val
    @prev = prev
  end
end
```

Программа для взаимодействия с пользователем:

```
# frozen_string_literal: true

require_relative 'my_math.rb'

# Class doc
class IOController
  class << self
    def start
      MyMath.calc_arctg(0.5, 0.0001).map { |i, k| puts i, k }
    end
  end
end

IOController.start
```

Тесты

```
# frozen_string_literal: true

require 'test/unit'
require_relative 'my_math'

# Class doc
class MyTest < Test::Unit::TestCase
  def setup
    # Do nothing
  end

  def teardown
    # Do nothing
  end

  def test1
    d = 0.0001
    val = Math.atan(0.5)
    assert_in_delta(MyMath.calc_arctg(0.5, d).map { |i| i }[-1], val, d)
  end

  def test2
    d = 0.001
    val = Math.atan(0.5)
    assert_in_delta(MyMath.calc_arctg(0.5, d).map { |i| i }[-1], val, d)
  end

  def test3
    d = 0.01
    val = Math.atan(0.5)
    assert_in_delta(MyMath.calc_arctg(0.5, d).map { |i| i }[-1], val, d)
  end
end
```

Часть 3

Составить метод scale отыскания масштаба графического изображения функции $f(x)$ на экране размером B единиц растра по формуле $M = \frac{B}{\max f(x)}$. В основной программе использовать метод для отыскания масштаба функций $x \cdot sin(x)$ и $tg(x)$, при $|x| < 1$.

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

Код основной программы:

```
# frozen_string_literal: true

# Class doc
class Scale
  class << self
    def scale(b_val, lambda = nil, &block)
      block = lambda unless block_given?
      b_val / block.call(1)
    end
  end
end
```

Программа для взаимодействия с пользователем:

```
# frozen_string_literal: true

require_relative 'scale'

# Class doc
class IOController
  class << self
    def main
      puts 'Введите B:'
      b = gets.to_f
      puts "Результат sin(x) * x: #{Scale.scale(b) { |x| x * Math.sin(x) }}"
      puts "Результат tg(x): #{Scale.scale(b, ->(x) { Math.tan(x) })}"
    end
  end
end

IOController.main
```

Тесты

```
# frozen_string_literal: true

require 'test/unit'
require_relative 'scale.rb'

# Test class
class MyTestPart3 < Test::Unit::TestCase
  def setup; end

  def test1
    assert_in_delta(5.94, Scale.scale(5) { |x| x * Math.sin(x) }, 0.1)
  end

  def test2
    assert_in_delta(3.21, Scale.scale(5, ->(x) { Math.tan(x) }), 0.1)
  end
end
```

```
 tetovske@pop-os  ~/YAIP/laby/laba6  study ●  ruby part1/test_part1.rb
Loaded suite part1/test_part1
Started
Введите количество и точность тестов:
5
0.001
-0.09369621419692875
-0.15899540830612857
-0.7099349826132644
-0.7084005627997196
-0.37654164272735113
.
Finished in 7.585730064 seconds.
----------------------------------------------------------------------------------------
1 tests, 5 assertions, 0 failures, 0 errors, 0 pendings, 0 omissions, 0 notifications
100% passed
----------------------------------------------------------------------------------------
0.13 tests/s, 0.66 assertions/s
 tetovske@pop-os  ~/YAIP/laby/laba6  study ●  ruby part2/part2_test.rb
Loaded suite part2/part2_test
Started
...
Finished in 0.000816571 seconds.
----------------------------------------------------------------------------------------
3 tests, 3 assertions, 0 failures, 0 errors, 0 pendings, 0 omissions, 0 notifications
100% passed
----------------------------------------------------------------------------------------
3673.90 tests/s, 3673.90 assertions/s
 tetovske@pop-os  ~/YAIP/laby/laba6  study ●  ruby part3/part3_test.rb
Loaded suite part3/part3_test
Started
..
Finished in 0.000506349 seconds.
----------------------------------------------------------------------------------------
2 tests, 2 assertions, 0 failures, 0 errors, 0 pendings, 0 omissions, 0 notifications
100% passed
----------------------------------------------------------------------------------------
3949.84 tests/s, 3949.84 assertions/s
 tetovske@pop-os  ~/YAIP/laby/laba6  study ●
```

Рис. 1 (Тесты программы)

```
 tetovske@pop-os  ~/YAIP/laby/laba6  study ●  rubocop *
Inspecting 9 files
.........

9 files inspected, no offenses detected
 tetovske@pop-os  ~/YAIP/laby/laba6  study ●
```

Рис. 2 (Отчёт Rubocop)

Вывод: *в ходе выполнения лабораторной работы было создано перечисление с помощью класса Enumerator, была изучена работа с блоками и лямбдами, а также с методами примеси Enumerable. Все программы были протестированы и проверены на соответствие стилю программой Rubocop.*