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

Дисциплина: Языки интернет-программирования

Студент	ИУ6-35 Б	27.10.2023	И.А. Дулина
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Е.Ю. Гаврилова
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2023

Вариант 8.

Цель работы: научиться использовать язык программирования Ruby, а именно Enumerator и Enumerable

Часть 1.

Задание:

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.

Код программы:

Файл пользователя (client.rb):

 # frozen_string_literal: true

```
require './main'
loop do
  puts('Введите радиус окружности: ')
  r = gets.chomp.to_f
  puts('Введите точность вычисления: ')
  puts('1) 0.001')
  puts('2) 0.0001')
  f = false
  n = 0
  loop do
    case gets.chomp.to_i
    when 1
      n = 0.001
      f = true
```

```
    when 2
      n = 0.0001
      f = true
    else
      puts('Вы ввели некорректное значение. Попробуйте снова')
    end
    break if f == true
  end
  puts('Число итераций: ')
  puts(i_am_matematic(n, r))
  break if gets.chomp == 'end'
end
```

Файл основной программы (main.rb):

```
# frozen_string_literal: true

def i_am_matematic(eps, rad)
  # s=1/2 R**2 n sin(2pi/n)
  n = 3
  iter = 0
  loop do
    n += 1
    iter += 1
    break if counting(n, rad) - counting(n - 1, rad) < eps
  end
  puts(iter - 1) # потому что последняя проверка не работает и просих выход из цикла
  puts(counting(n - 1, rad))
  puts(counting(n, rad))
  puts(counting(n + 1, rad))
```

```
  counting(n - 1, rad).round(4).to_f
end

# number - количество сторон
def counting(number, rad)
  0.5 * rad * rad * number * Math.sin(2 * Math::PI / number)
end
```

Файл тестов (test.rb):

```
# frozen_string_literal: true

require 'test/unit'
require './main'

# test
class TestPal < Test::Unit::TestCase
  def test_pal
    assert_in_delta(Math::PI * 5 * 5, i_am_matematic(0.001, 5), 0.1)
    assert_in_delta(Math::PI * 3 * 3, i_am_matematic(0.0001, 3), 0.1)
    assert_in_delta(Math::PI * 2 * 2, i_am_matematic(0.001, 2), 0.1)
  end
end
```

```
Введите радиус окружности:
3
Введите точность вычисления:
1) 0.001
2) 0.0001
2
Число итераций:
152
28.2666
```

```
Введите радиус окружности:
3
Введите точность вычисления:
1) 0.001
2) 0.0001
1
Число итераций:
69
28.2385
```

Рисунки 1.1 и 1.2 – результаты работы программы

При увеличении точности число итераций увеличивается

```
##teamcity[testFinished name = 'test_pal' duration = '2' nodeId = 'TestPal.test_pal' timestamp = '2023-10-26T19:52:45.536+0300']
##teamcity[testSuiteFinished name = 'TestPal' nodeId = 'TestPal' timestamp = '2023-10-26T19:52:45.536+0300']
1 tests, 3 assertions, 0 failures, 0 errors, 0 pendings, 0 omissions, 0 notifications
Test suite finished: 0.0024354 seconds

Process finished with exit code 0
```

Рисунок 1.3 – результат работы тестов

```
PS D:\education\3 semester\ipl\lab\lab6\lab6_1> rubocop client.rb
Inspecting 1 file
C

Offenses:

client.rb:30:1: C: [Correctable] Style/BlockComments: Do not use block comments.
=begin ...
^^^^^^
client.rb:43:5: C: [Correctable] Layout/TrailingEmptyLines: Final newline missing.
=end


1 file inspected, 2 offenses detected, 2 offenses autocorrectable
```

Рисунок 1.4 – результат работы rubocop client.rb

```
PS D:\education\3 semester\ipl\lab\lab6\lab6_1> rubocop main.rb
Inspecting 1 file
.

1 file inspected, no offenses detected
```

Рисунок 1.5 – результат работы rubocop main.rb

```
PS D:\education\3 semester\ipl\lab\lab6\lab6_1> rubocop test.rb
Inspecting 1 file
.

1 file inspected, no offenses detected
```

Рисунок 1.6 – результат работы rubocop test.rb

Часть 2.

Задание:

Решить предыдущее задание с помощью Enumerable или Enumerator.

Код программы:

Файл пользователя (client.rb):

 # frozen_string_literal: true

```
require './main'
loop do
  puts('Введите радиус окружности: ')
  r = gets.chomp.to_f
  puts('Введите точность вычисления: ')
  puts('1) 0.001')
  puts('2) 0.0001')
  f = false
  n = 0
  loop do
    case gets.chomp.to_i
    when 1
      n = 0.001
      f = true
    when 2
      n = 0.0001
      f = true
    else
```

```
    puts('Вы ввели некорректное значение. Попробуйте снова')
   end
   break if f == true
  end
  puts('Число итераций: ')
  puts(i_am_matematic(n, r))
  break if gets.chomp == 'end'
end
```

Файл основной программы (main.rb):

```
# frozen_string_literal: true

def i_am_matematic(eps, rad)
  list = Enumerator.new do |yielder|
    number = 3
    sum = counting(number + 1, rad)
    prev = counting(number, rad)
    iter = 0
    loop do
      yielder.yield sum, prev, iter
      prev = sum
      number += 1
      sum = counting(number + 1, rad)
      iter += 1
    end
    puts(iter)
  end
  a = list.find { |sum, prev| (prev - sum).abs < eps }.each
  a.next # число которое уже не подходит по условию
```

```
  summa = a.next.round(4).to_f
  iter = a.next
  puts(iter)
  summa
end

# number - количество сторон
def counting(number, rad)
  0.5 * rad * rad * number * Math.sin(2 * Math::PI / number)
end
```

Файл тестов (test.rb):

```
# frozen_string_literal: true

require 'test/unit'
require './main'

# test
class TestPal < Test::Unit::TestCase
  def test_pal
    assert_in_delta(Math::PI * 5 * 5, i_am_matematic(0.001, 5), 0.1)
    assert_in_delta(Math::PI * 3 * 3, i_am_matematic(0.0001, 3), 0.1)
    assert_in_delta(Math::PI * 2 * 2, i_am_matematic(0.001, 2), 0.1)
  end
end
```

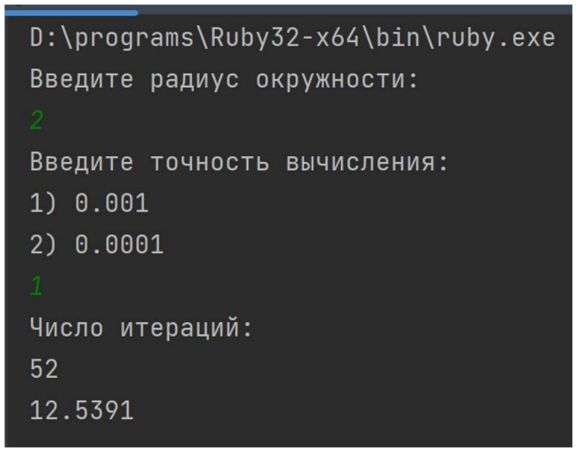

Рисунок 2.1 – результат работы программы

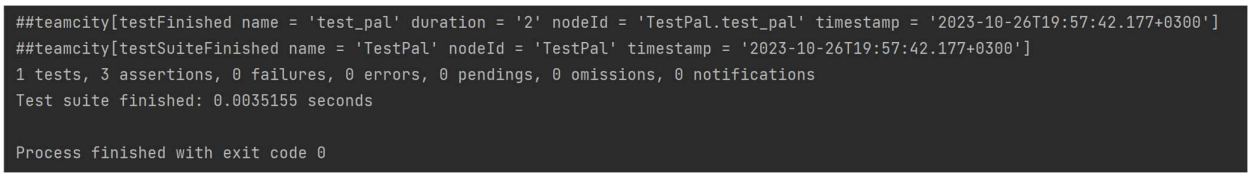

Рисунок 2.2 – результат работы тестов

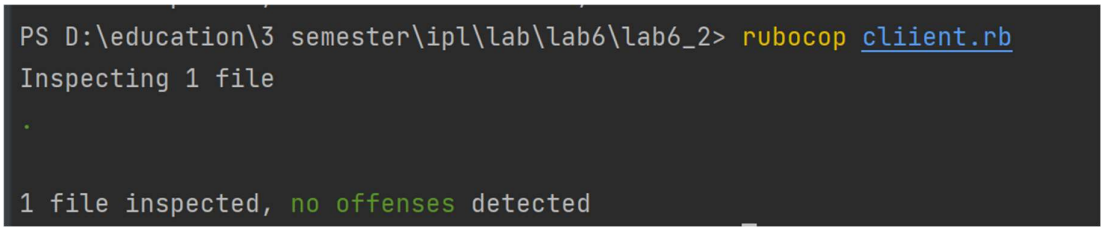

Рисунок 2.3 – результат работы rubocop client.rb

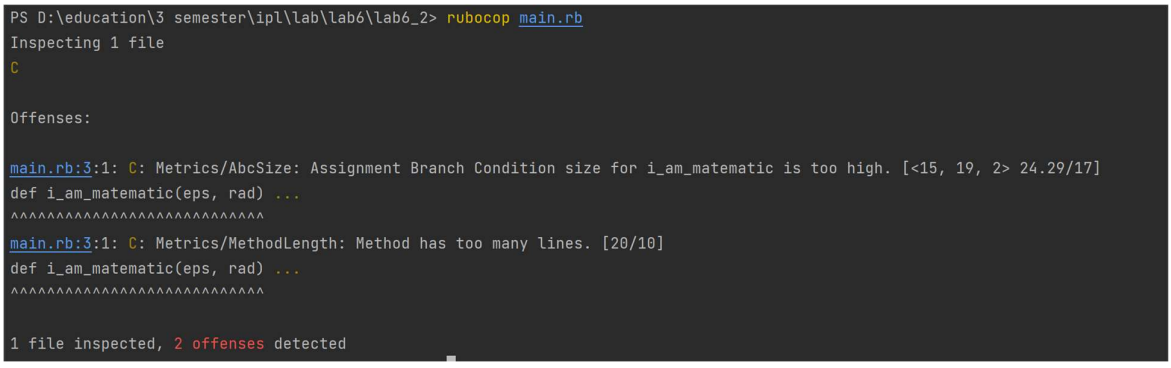

Рисунок 2.4 – результат работы rubocop main.rb

```
PS D:\education\3 semester\ipl\lab\lab6\lab6_2> rubocop test.rb
Inspecting 1 file
.

1 file inspected, no offenses detected
```

Рисунок 2.5 – результат работы test.rb

Часть 3.

Задание:

Составить метод minmax, отыскивающую $x \in [a, b]$, для которого функция $y = f(x)$ принимает максимальное и минимальное значение с точностью $0,01$. В основной программе использовать этот метод для математических функций $y = \frac{x-1}{x+2}; x \in [0, 2]$ и $y = sin(\frac{x}{2} - 1), x \in [-1, 1]$.

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

Код программы:

Файл пользователя (client.rb):

 # frozen_string_literal: true

```
require './main'
loop do
  puts('Выберите функцию: ')
  puts('1) y = (x-1)/(x+2)')
  puts('2) y = sin(x/2 - 1)')
  f = false
  loop do
    case gets.chomp.to_i
    when 1
      f = true
```

```
    stt = 0.0
    enn = 2.0
    puts('Вызов через блок')
    puts(minmax(stt, enn) { |x| (x - 1) / (x + 2) }) # через блок
    puts('Вызов через лямбда')
    lb = ->(x) { (x - 1) / (x + 2) } # лямбда выражение
    puts(minmax(stt, enn, &lb))
   when 2
    f = true
    stt = -1.0
    enn = 1.0
    puts('Вызов через блок')
    puts(minmax(stt, enn) { |x| Math.sin(x / 2 - 1) }) # через блок
    puts('Вызов через лямбда')
    lb = ->(x) { Math.sin(x / 2 - 1) } # лямбда выражение
    puts(minmax(stt, enn, &lb))
   else
    puts('Вы ввели некорректное значение. Попробуйте снова')
   end
   break if f == true
  end
  break if gets.chomp == 'end'
end
```

Файл основной программы (main.rb):

```
# frozen_string_literal: true

def minmax(stt, enn, &block)
  eps = 0.01
```

```
  stt.to_f
  enn.to_f
  massive = {}
  steps = ((enn - stt) / eps).truncate + 1 # количество шагов
  steps.times do
    massive[stt.round(3)] = block.call(stt.round(3)).round(4).to_f
    stt += eps
  end
  extrem = []
  extrem << massive.min_by(&:last).first
  extrem << massive.max_by(&:last).first
  extrem
end
```

Файл тестов (test.rb):

```
# frozen_string_literal: true

require 'test/unit'
require './main'

# test
class TestMinMax < Test::Unit::TestCase
  def test1
    lb = ->(x) { Math.sin(x * x - 1) }
    @res = minmax(0, 2, &lb)[1].to_f
    assert_in_delta(1.6, @res, 0.01)
  end

  def test2
```

```
    lb = ->(x) { x * x * x + x * x }
    @res = minmax(-2, 0, &lb)[1].to_f
    assert_in_delta(-0.67, @res, 0.01)
  end

  def test3
    @res = minmax(1, 3) { |x| -1 / (x * x) }[1]
    assert_in_delta(3, @res, 0.01)
  end

  def test4
    @res = minmax(0, 2) { |x| Math.cos(x * x + 3 * x - 1) }[0] # минимум
    assert_in_delta((1.0 / 2) * Math.sqrt(13 + 4 * Math::PI) - 3.0 / 2, @res, 0.01)
  end
end
```

```
Выберите функцию:
1) y = (x-1)/(x+2)
2) y = sin(x/2 - 1)
2
Вызов через блок
-1.0
1.0
Вызов через лямбда
-1.0
1.0
```

Рисунок 3.1 – результат работы программы

```
4 tests, 4 assertions, 0 failures, 0 errors, 0 pendings, 0 omissions, 0 notifications
Test suite finished: 0.0058068 seconds

Process finished with exit code 0
```

Рисунок 3.2 – результат работы тестов

```
PS D:\education\3 semester\ipl\lab\lab6\lab6_3> rubocop client.rb
Inspecting 1 file
C

Offenses:

client.rb:4:1: C: Metrics/BlockLength: Block has too many lines. [30/25]
loop do ...
^^^^^^^

1 file inspected, 1 offense detected
```

Рисунок 3.3 – результат работы client.rb

```
PS D:\education\3 semester\ipl\lab\lab6\lab6_3> rubocop main.rb
Inspecting 1 file
C

Offenses:

main.rb:3:1: C: Metrics/AbcSize: Assignment Branch Condition size for minmax is too high. [<6, 19, 2> 20.02/17]
def minmax(stt, enn, &block) ...
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
main.rb:3:1: C: Metrics/MethodLength: Method has too many lines. [13/10]
def minmax(stt, enn, &block) ...
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

1 file inspected, 2 offenses detected
```

Рисунок 3.4 – результат работы main.rb

```
PS D:\education\3 semester\ipl\lab\lab6\lab6_3> rubocop test.rb
Inspecting 1 file
.

1 file inspected, no offenses detected
```

Рисунок 3.5 – результат работы test.rb

Вывод: были изучены возможности передавать в функцию блок программы, а также реализовывать функцию при помощи Enumerator или Enumerable.