



**«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

## **О т ч е т**

**по лабораторной работе № 6**

**Дисциплина:** Языки Интернет-программирования

Студент гр. ИУ6-35Б **Коньшин Н.М.**

## Лабораторная работа №6

### Задание №1:

Решить задачу с точностью  $\xi = 10^{-2}, 10^{-4}$ , организовав итерационный n

цикл. Найти первый член последовательности  $y = \frac{n}{n^2 + 2}$ , для которого  $y < \xi$ . Определить, как изменяется число итераций при изменении точности.

Текст программы для взаимодействия с пользователем:

```
require_relative 'logic_lab6.rb'
puts 'Полученное значение:'
e1 = 0.01
e2 = 0.001
print "Значение при e = ", e1
puts
puts schet(e1)
print "Значение при e = ", e2
puts
puts schet(e2)
```

Текст основной программы:

```
def schet(e)
  n = 1.0
  k = 0
  y = n / (n * n + 2)
  while y >= e
    n += 1
    y = n / (n * n + 2)
    k += 1
  end
  y
end
```

Текст тестовой программы:

```
require 'test/unit'
require_relative 'lab_6.rb'

class FooTest < Test::Unit::TestCase
  def test_001
    assert_equal 0.009998000399920015, schet(0.01)
  end

  def test_002
    assert_equal 0.000999998000004, schet(0.001)
  end
end
```

Результат работы (Рис. 1):

```

[nikita@MBPro13MK Часть 1 % ruby lab6_p1_forUser.rb
Полученное значение:
Значение при e = 0.01
0.009998000399920015
Значение при e = 0.001
0.000999998000004
nikita@MBPro13MK Часть 1 % █

```

Рис. 1

Результат работы тестов (Рис. 2):

```

Loaded suite lab6_p1_MiniTest
Started
..
Finished in 0.000626 seconds.
-----
2 tests, 2 assertions, 0 failures, 0 errors, 0 pendings, 0 omissions, 0 notifications
100% passed
-----
3194.89 tests/s, 3194.89 assertions/s
nikita@MBPro13MK Часть 1 % █

```

Рис. 2

## Задание №2:

Решить предыдущее задание с помощью Enumerator.

Текст программы для взаимодействия с пользователем:

```

require_relative 'logic_lab6.rb'
puts 'Полученное значение:'
e1 = 0.01
e2 = 0.0001
a = 0.0
a = schet(e1)
b = schet(e2)
puts a
puts b

```

Текст основной программы:

```

def schet(e)
  en = Enumerator.new do |y|
    n = 1.0
    y1 = n / (n*n+2)
    loop do
      y.yield y1
      n = n + 1
      y1 = n / (n*n+2)
    end
  end
  en.find { |x| x < e }
end

```

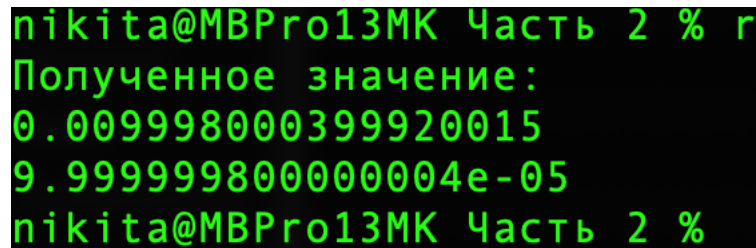
Текст тестовой программы:

```
require 'test/unit'
require_relative 'lab_6.rb'

class FooTest < Test::Unit::TestCase
  def test_001
    assert_equal 0.009998000399920015, schet(0.01)
  end

  def test_002
    assert_equal 0.0009999980000004, schet(0.001)
  end
end
```

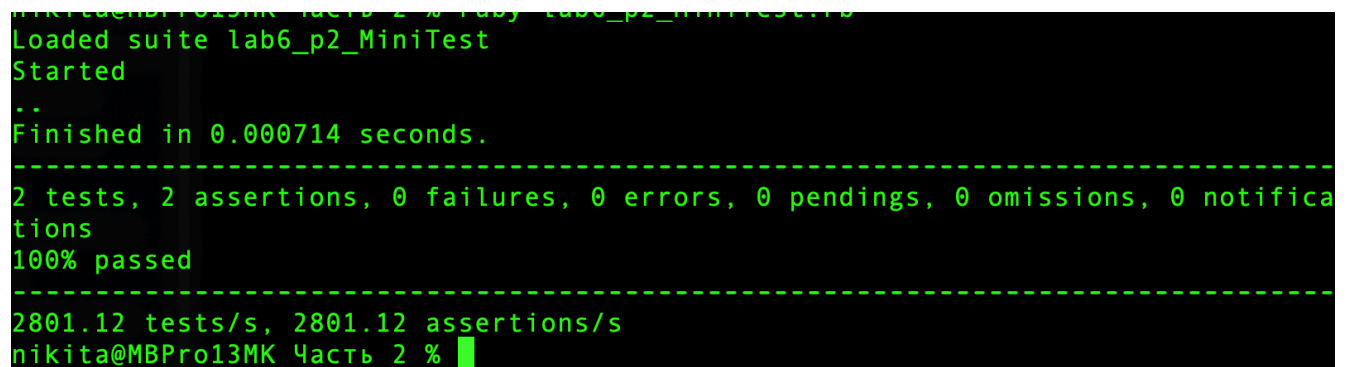
Результат работы программы (Рис. 3):



```
nikita@MBPro13MK Часть 2 % r
Полученное значение:
0.009998000399920015
9.999998000000004e-05
nikita@MBPro13MK Часть 2 %
```

Рис. 3

Результат работы тестов (Рис. 4):



```
nikita@MBPro13MK Часть 2 % ruby lab6_p2_MiniTest.rb
Loaded suite lab6_p2_MiniTest
Started
..
Finished in 0.000714 seconds.
-----
2 tests, 2 assertions, 0 failures, 0 errors, 0 pendings, 0 omissions, 0 notifications
100% passed
-----
2801.12 tests/s, 2801.12 assertions/s
nikita@MBPro13MK Часть 2 % █
```

Рис. 4

### Задание №3:

Составить метод differ для вычисления производных функции  $Y(X)$  в некоторых 3 соседних точках, отстоящих на величину шага  $h$ . Для вычислений использовать формулы

Лагранжа:  $y'_0 = \frac{-3y_0 + 4y_1 - y_2}{2h}$ ;  $y'_1 = \frac{-y_0 + y_2}{2h}$ ;  $y'_2 = \frac{y_0 - 4y_1 + 3y_2}{2h}$ , где  $y_0, y_1$  и  $y_2$  – координаты

точек. В основной программе использовать метод differ для вычисления производных функций  $\sin(x)$  и  $\lg(x + 1)$  в точках 0.49, 0.5 и 0.51.

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

Текст программы для взаимодействия с пользователем:

```
require_relative 'logic_lab6.rb'
puts 'Выберите функцию: sin - 1, tg - 2'
n = gets.to_i
proisv = Proisvodnaya.new(n)
```

```

proc_1 = lambda { |x,y,z| puts 'Полученные значения производных: ', x,y,z}

proisv.visov{|x,y,z| puts ' Полученные значения производных: ', x , y , z }
proisv.visov(proc_1)

```

### Текст основной программы:

```

class Proisvodnaya
  def initialize(n)
    @n = n
    if @n == 1 then
      y0 = Math.sin(0.49)
      y1 = Math.sin(0.5)
      y2 = Math.sin(0.51)
    else
      y0 = Math.sin(0.49+1)/Math.cos(0.49+1)
      y1 = Math.sin(0.5+1)/Math.cos(0.5+1)
      y2 = Math.sin(0.51+1)/Math.cos(0.51+1)
    end
    differ(y0,y1,y2)
  end

  def differ(y0,y1,y2)
    h = 0.01
    @y01 = ((-1)*3*y0+4*y1-y2)/(2*h)
    @y11 = ((-1)*y0+y2)/(2*h)
    @y21 = (y0-4*y1+3*y2)/(2*h)
    a = @y01+@y11+@y21
    puts a
    return a
  end

  def visov(lam = nil)
    if block_given?
      yield(@y01,@y11,@y21)
    elsif lam
      lam.call(@y01,@y11,@y21)
    end
  end
end

```

### Текст тестовой программы:

```

require 'test/unit'
require_relative 'logic_lab6.rb'

class FooTest < Test::Unit::TestCase
  def test_001
    proisv = Proisvodnaya.new(1)
    assert_equal 2.6327038067624264,
    proisv.differ(0.470625888171158,0.479425538604203,0.48817724688290753)
  end

  def test_002
    proisv = Proisvodnaya.new(2)
    assert_equal 611.7352893389303,
    proisv.differ(12.349856441625802,14.10141994717172,16.428091703885336)
  end
end

```

Результат работы программы (Рис. 5):

```
Выберите функцию: sin - 1, tg - 2
2
611.7352893389303
Полученные значения производных:
146.40093799620715
203.9117631129767
261.42258822974645
Полученные значения производных:
146.40093799620715
203.9117631129767
261.42258822974645
nikita@MBPro13MK Часть 3 %
```

Рис. 5

Результат работы тестов (Рис. 6):

```
Loaded suite lab6_p3_MiniTest
Started
2.6327038067624264
2.6327038067624264
.611.7352893389303
611.7352893389303
.
Finished in 0.000558 seconds.
-----
2 tests, 2 assertions, 0 failures, 0 errors, 0 pendings, 0 omissions, 0 notifications
100% passed
-----
3584.23 tests/s, 3584.23 assertions/s
nikita@MBPro13MK Часть 3 %
```

Рис. 6

Вывод: в данной лабораторной работе был создан код метода для расчета значения производных и программы для взаимодействия с пользователем. Метод может принимать на вход lambda-выражения и блоки. Создан код тестирующей программы. Проведено тестирование. Тестирование показало корректность работы программы. Все файлы прошли валидацию программой Rubocop.