|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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:** Создание консольного приложения Ruby.

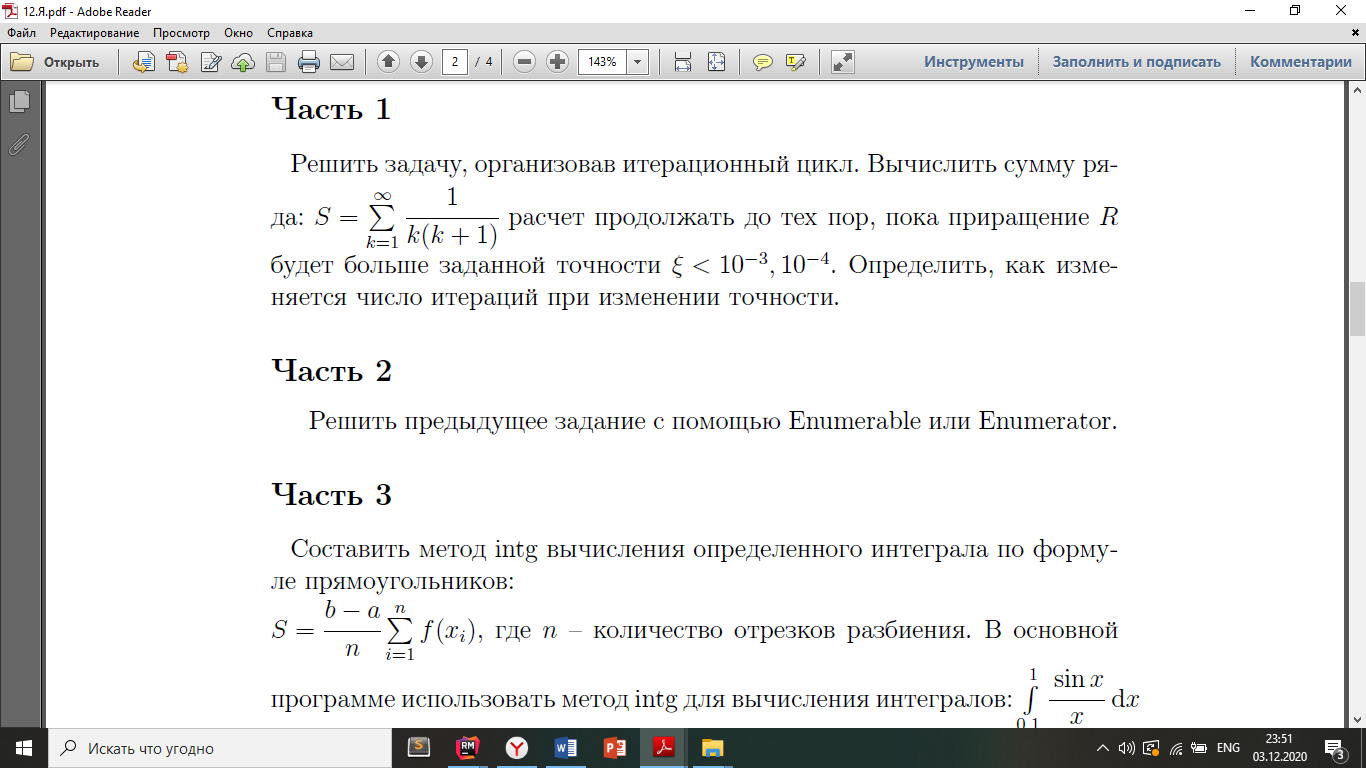
**Дисциплина:** Языки интернет-программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6 |  |  |  |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Лабораторная работа №6**

**Часть 1:**



**Код в console.rb:**

*# frozen\_string\_literal: true  
  
require\_relative* 'logic'  
  
puts "eps = 1e-3: res = #{find(1e-3)}"  
puts "eps = 1e-4: res = #{find(1e-4)}"

**Код в logic.rb:**

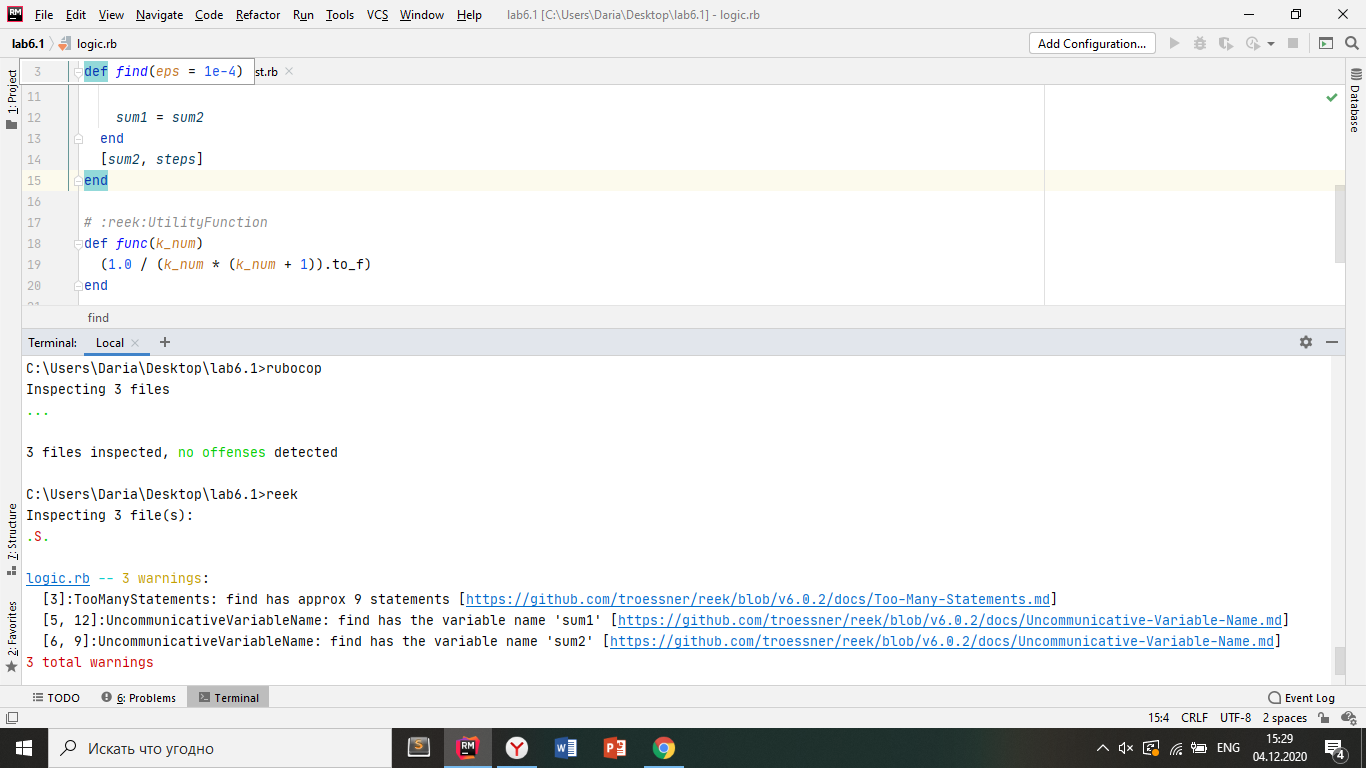
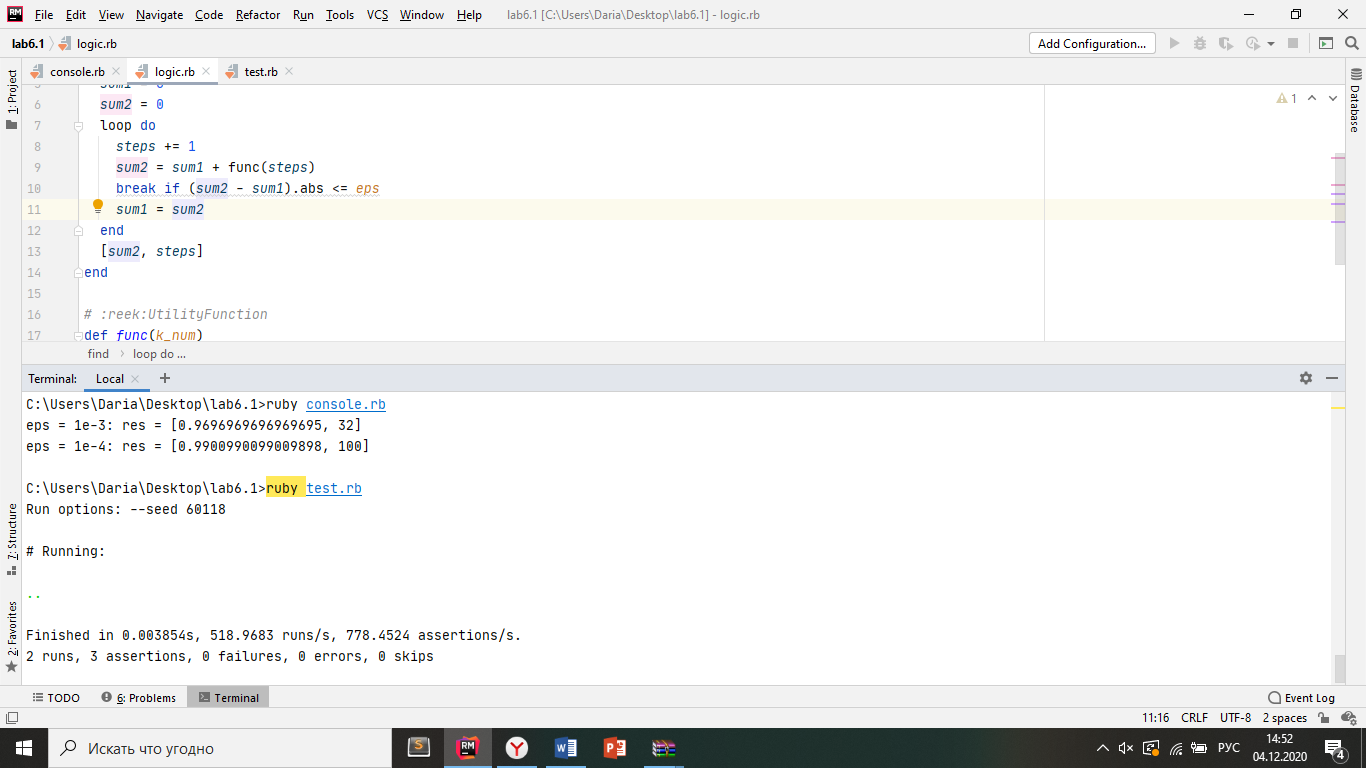
*# frozen\_string\_literal: true*def *find*(*eps* = 1e-4)  
 *steps* = 0  
 *sum1* = 0  
 *sum2* = 0  
 loop do  
 *steps* += 1  
 *sum2* = *sum1* + func(*steps*)  
 break if (*sum2* - *sum1*).abs <= *eps*

*sum1* = *sum2* end  
 [*sum2*, *steps*]  
end  
  
*# :reek:UtilityFunction*def *func*(*k\_num*)  
 (1.0 / (*k\_num* \* (*k\_num* + 1)).to\_f)  
end

**Код в test.rb:**

*# frozen\_string\_literal: true  
  
require* 'minitest/autorun'  
*require* 'minitest/assertions'  
*require\_relative* 'logic'  
  
*#* ***class*** *for test*class *MyTest* < *Minitest*::*Test* def *test\_first  
 result* = find(1e-3)  
 assert\_in\_delta(0.9696969696969695, *result*[0], 1e-3)  
 assert\_equal(32, *result*[1])  
 end  
  
 def *test\_second  
 result* = find(1e-4)  
 assert\_in\_delta(0.9900990099009898, *result*[0], 1e-4)  
 end  
end

**Результат выполнения и результаты проверки анализаторами rubocop и reek:**



**Часть 2:**

Решить предыдущее задание с помощью Enumerator и Enumerable

**Код в console.rb:**

*# frozen\_string\_literal: true  
  
require\_relative* 'logic'  
  
puts "eps = 1e-3: res = #{answer(1e-3)[0]}"  
puts "eps = 1e-4: res = #{answer(1e-4)[0]}"

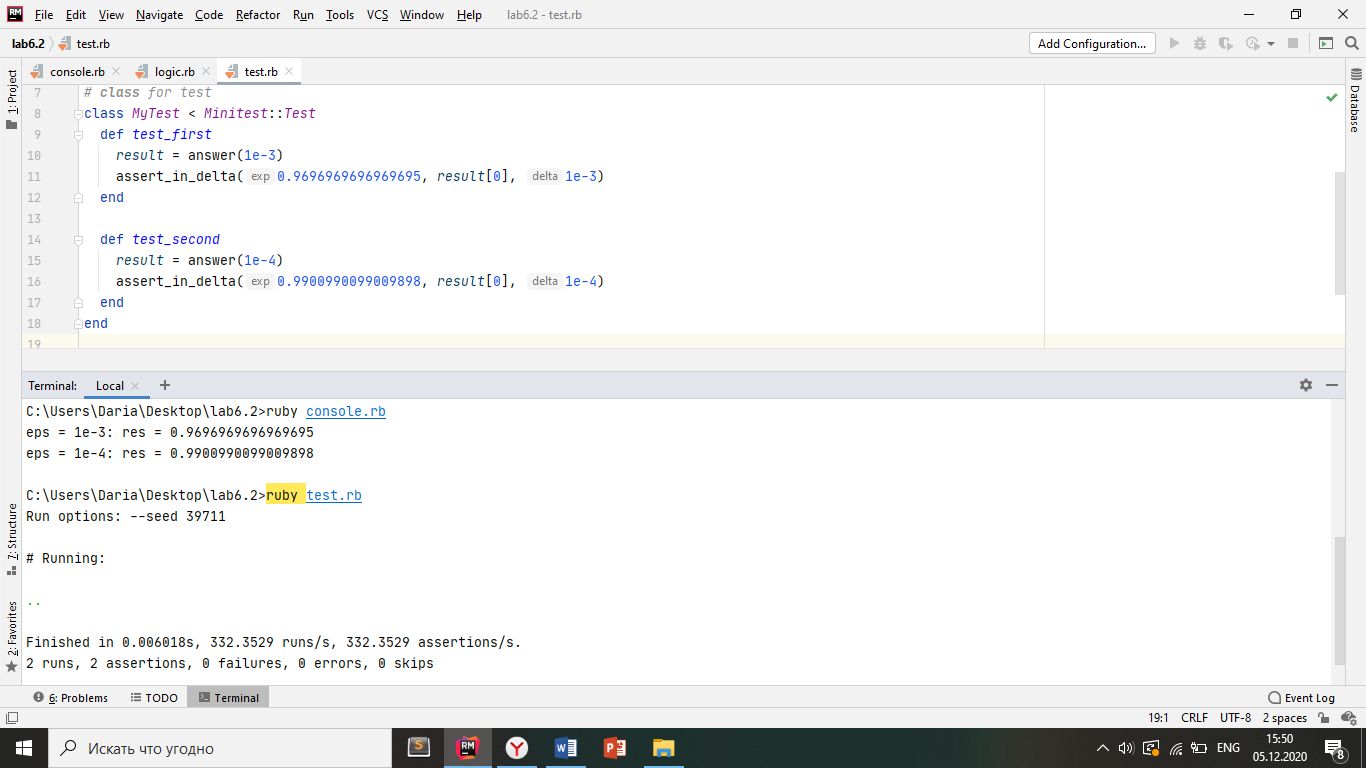
**Код в logic.rb:**

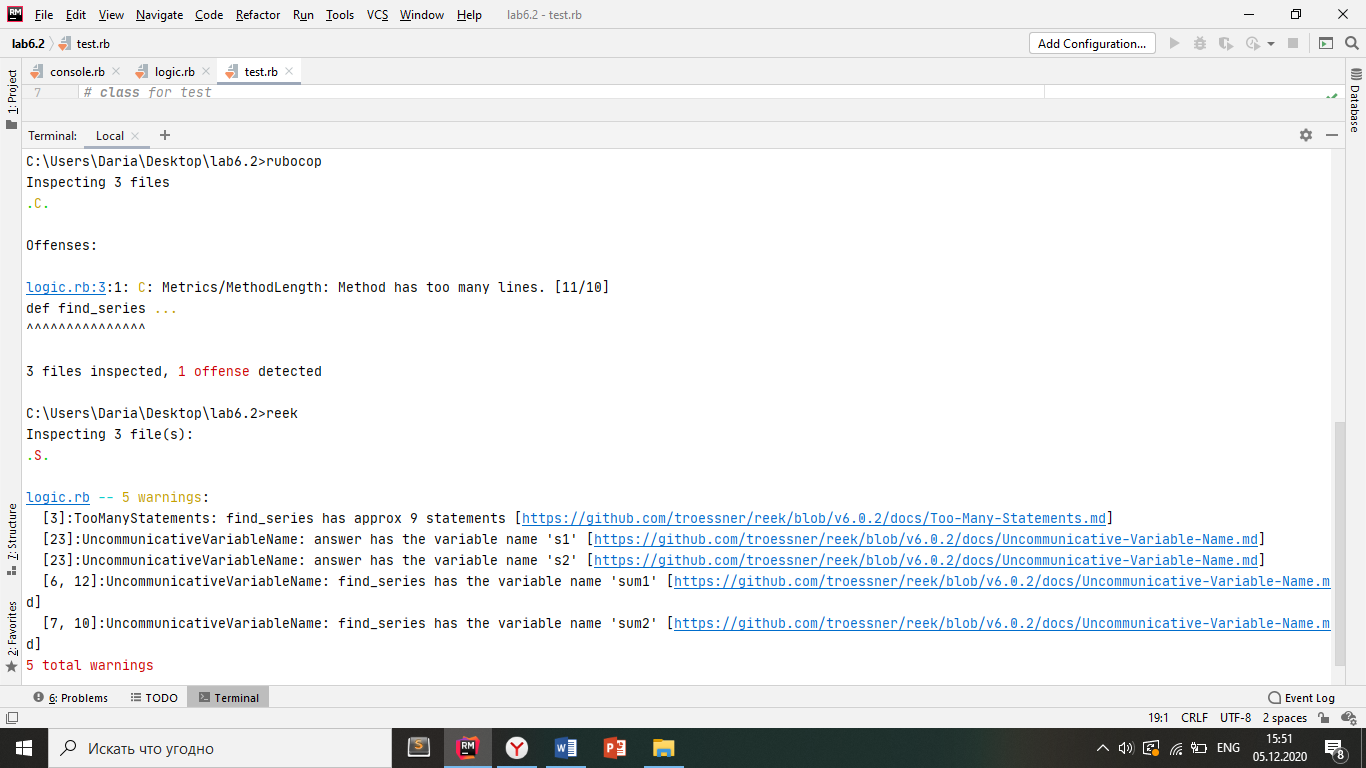
*# frozen\_string\_literal: true*def *find\_series  
 Enumerator*.new do |*yielder*|  
 *steps* = 0  
 *sum1* = 0  
 *sum2* = 0  
 loop do  
 *steps* += 1  
 *sum2* += func(*steps*)  
 *yielder* << [*sum2*, *sum1*]  
 *sum1* = *sum2* end  
 end  
end  
  
*# :reek:UtilityFunction*def *func*(*k\_num*)  
 (1.0 / (*k\_num* \* (*k\_num* + 1)).to\_f)  
end  
  
def *answer*(*eps*)  
 find\_series.find { |*s2*, *s1*| (*s2* - *s1*).abs <= *eps* }  
end

**Код в test.rb:**

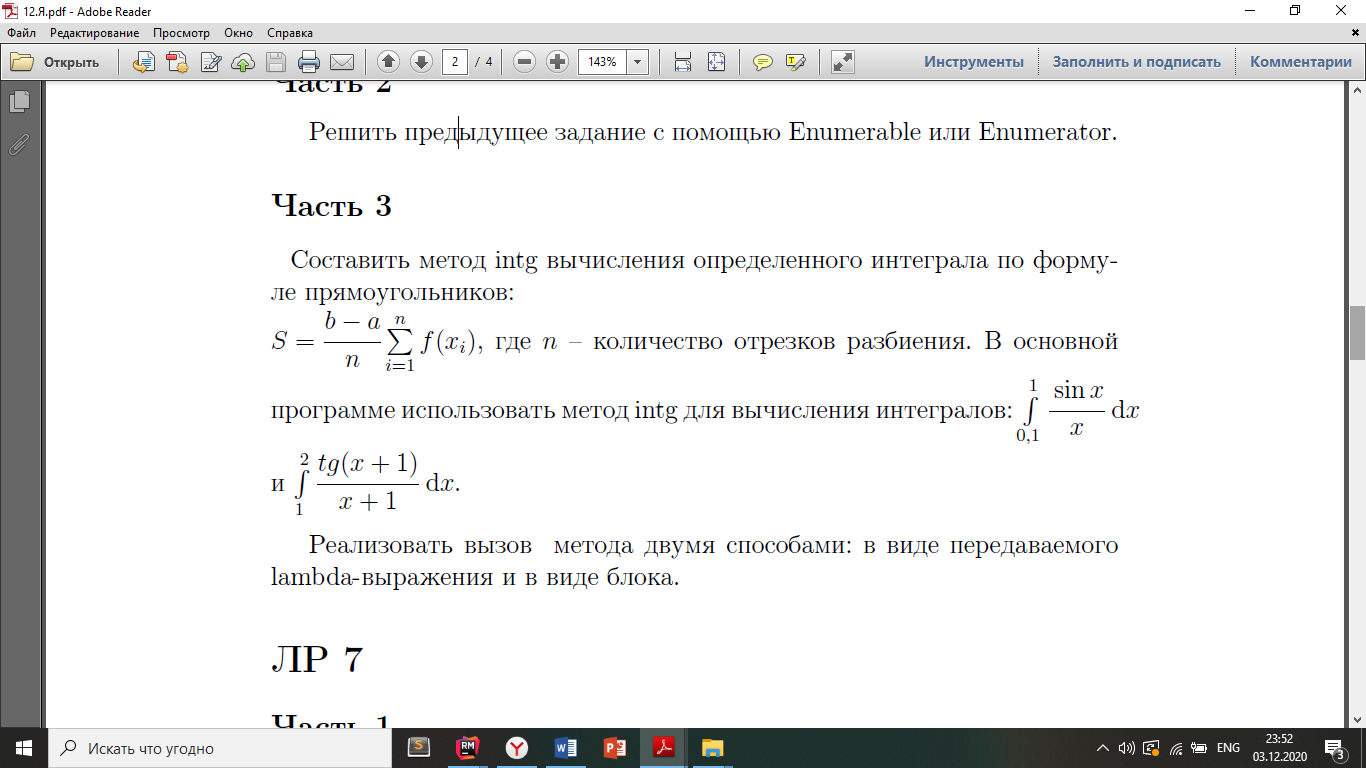
*# frozen\_string\_literal: true  
  
require* 'minitest/autorun'  
*require* 'minitest/assertions'  
*require\_relative* 'logic'  
  
*#* ***class*** *for test*class *MyTest* < *Minitest*::*Test* def *test\_first  
 result* = answer(1e-3)  
 assert\_in\_delta(0.9696969696969695, *result*[0], 1e-3)  
 end  
  
 def *test\_second  
 result* = answer(1e-4)  
 assert\_in\_delta(0.9900990099009898, *result*[0], 1e-4)  
 end  
end

**Результат выполнения и результаты проверки анализаторами rubocop и reek:**





**Часть 3:**



**Код в console.rb:**

*# frozen\_string\_literal: true  
  
require\_relative* 'logic'  
  
puts 'Choose a function:'  
puts '1) sin(x) / x'  
puts '2) tg(x + 1) / (x + 1)'  
*code* = gets.to\_i  
  
puts 'Input step'  
*steps* = gets.to\_i  
*func* = ->(*x*) { *Math*.sin(*x*) / *x* }  
case *code*when 1  
 puts intg(*steps*, 1, 0.1, *func*)  
when 2  
 puts intg(*steps*, 2, 1) { |*x*| *Math*.tan(*x* + 1) / (*x* + 1) }  
else puts 'error'  
end

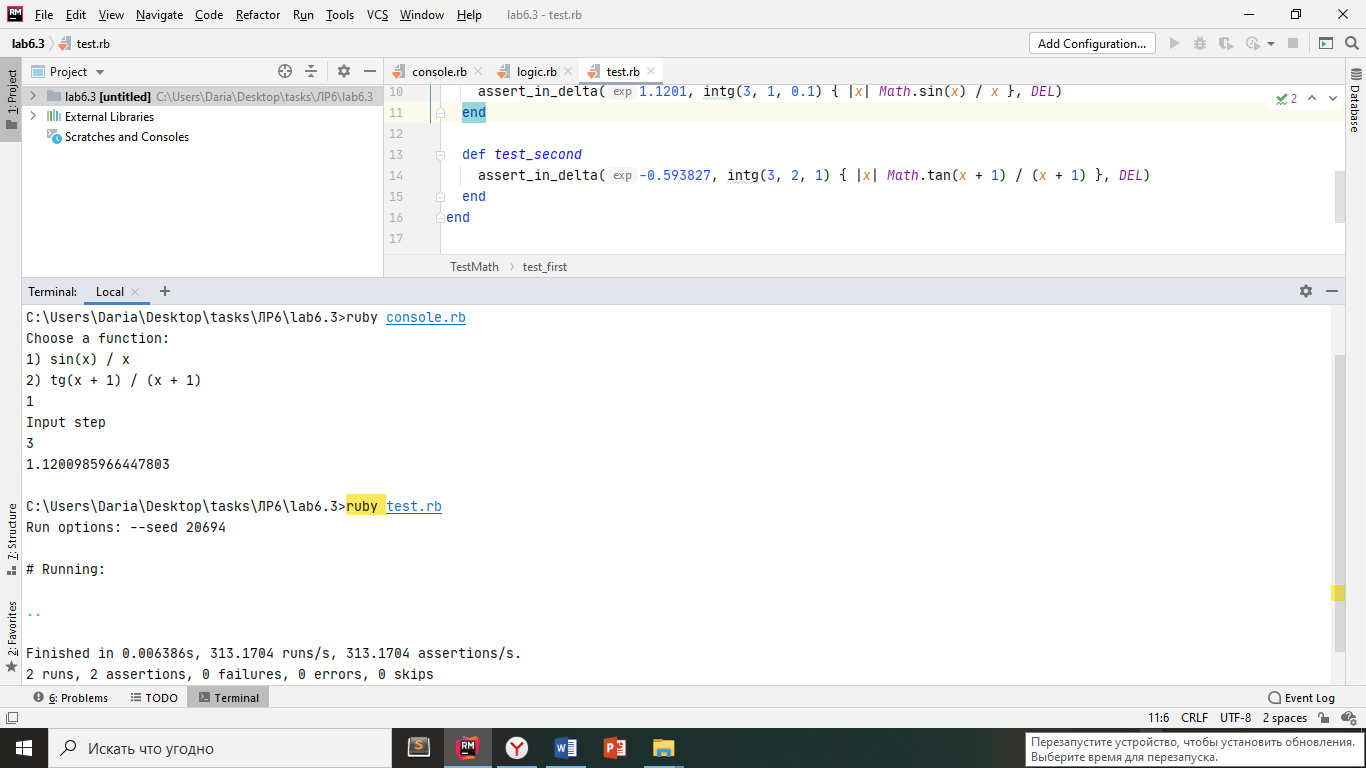
**Код в logic.rb:**

*# frozen\_string\_literal: true*def *intg*(*step*, *top*, *bottom*, *func* = false)  
 *square* = 0  
 *d\_x* = (*top* - *bottom*) / *step*.to\_f  
 (*bottom*..*top*).step(*d\_x*) { |*cur\_pos*| *square* += *func* ? *func*.call(*cur\_pos*) : yield(*cur\_pos*) }  
 *square* \* *d\_x*end

**Код в test.rb:**

*# frozen\_string\_literal: true  
  
require* 'minitest/autorun'  
*require\_relative* 'logic'  
  
*#* ***class*** *with test*class *TestMath* < *Minitest*::*Test  
 DEL* = 1e-2  
 def *test\_first* assert\_in\_delta(1.1201, intg(3, 1, 0.1) { |*x*| *Math*.sin(*x*) / *x* }, *DEL*)  
 end  
  
 def *test\_second* assert\_in\_delta(-0.593827, intg(3, 2, 1) { |*x*| *Math*.tan(*x* + 1) / (*x* + 1) }, *DEL*)  
 end  
end

**Результат выполнения и результаты проверки анализаторами rubocop и reek:**



****

**Вывод:** **в ходе данной лабораторной работы я изучила основные принципы программирования на языке Ruby.**