|  |  |
| --- | --- |
| 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 **09.03.03 Прикладная информатика**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 6 |

**Дисциплина:** Языки интернет-программирования



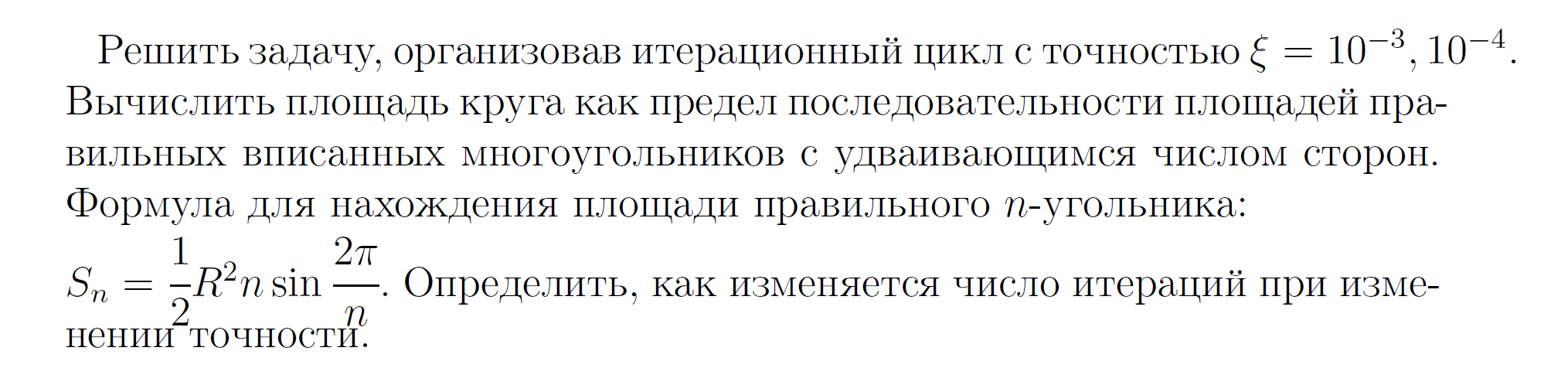
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-35 Б |  | 27.10.2023 | И.А. Дулина |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Е.Ю. Гаврилова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Вариант 8.**

**Часть 1.**

**Задание:**



**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

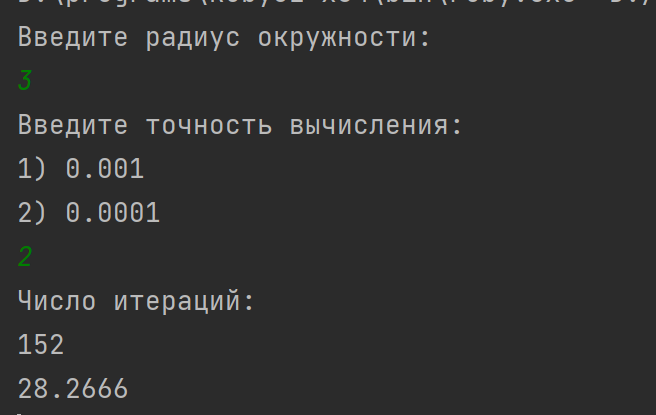
*Файл основной программы (main.rb):*

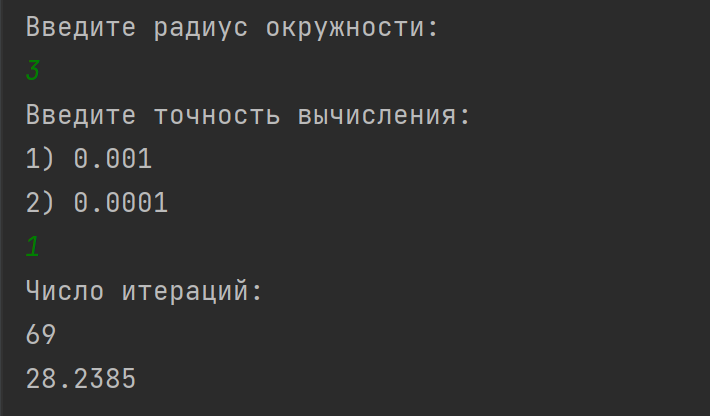
*# frozen\_string\_literal: true*def i\_am\_matematic(*eps*, *rad*)  
 *# s=1/2 R\*\*2 n sin(2pi/n)  
 n* = 3  
 *iter* = 0  
 loop do  
 *n* += 1  
 *iter* += 1  
 break if counting(*n*, *rad*) - counting(*n* - 1, *rad*) < *eps* end  
 puts(*iter* - 1) *# потому что последняя проверка не работает и 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

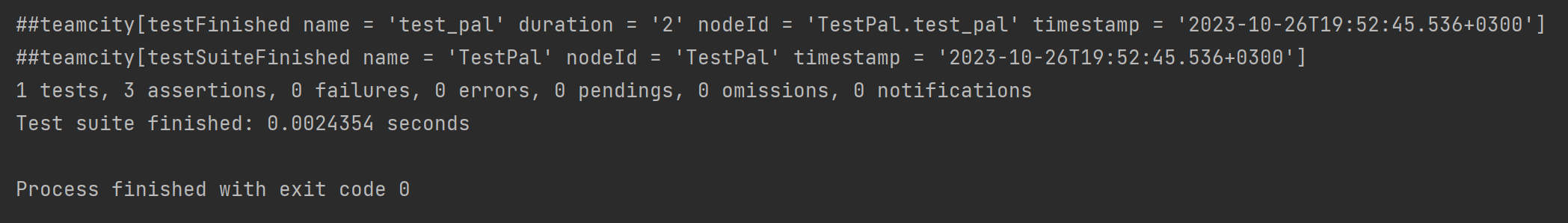
**Результат работы программы:**

****

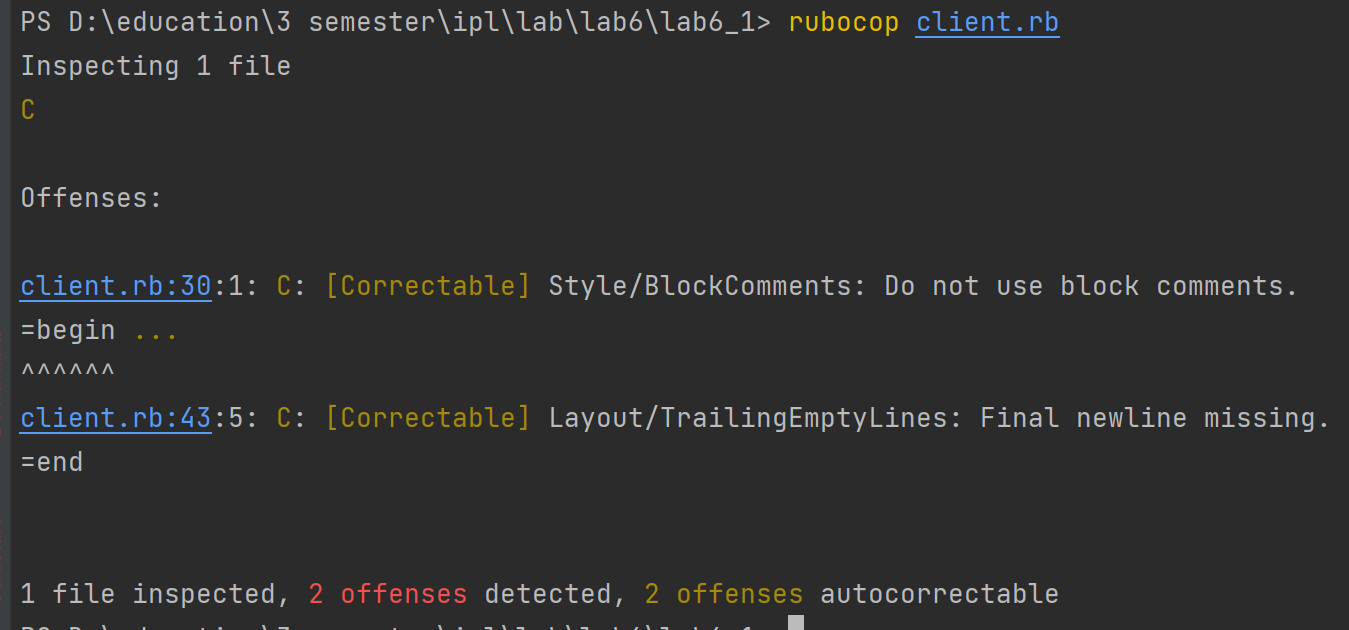
****

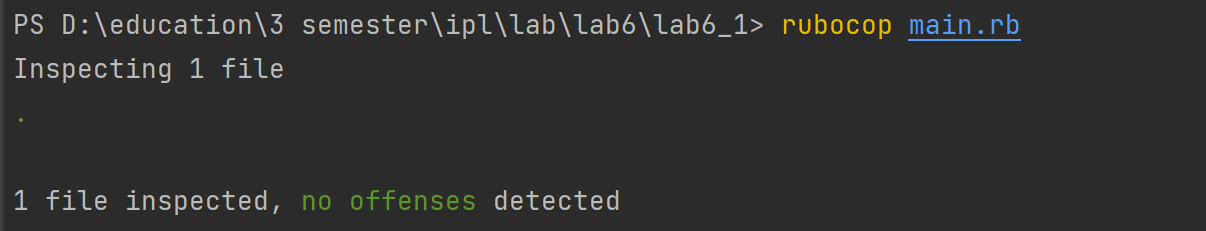
При увеличении точности число итераций увеличивается

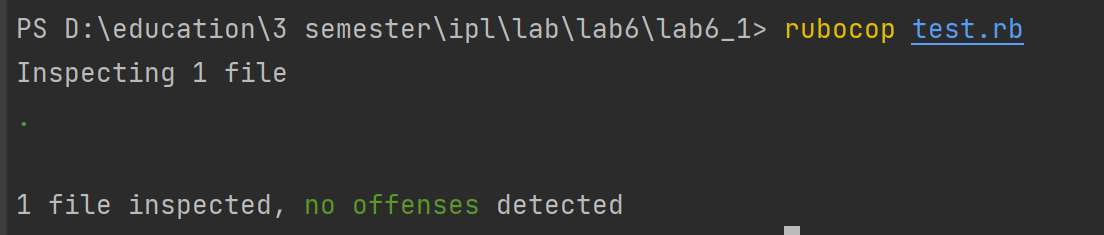
**Результат работы тестов:**

****

**Результат работы rubocop:**

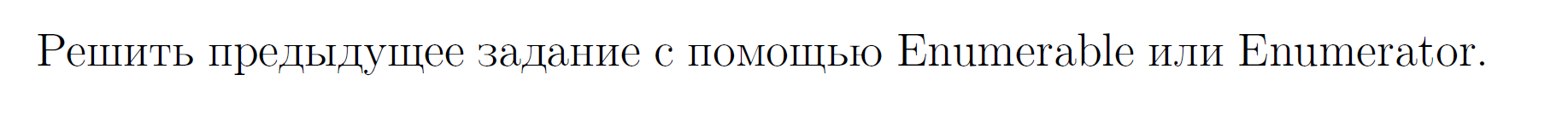
****





**Часть 2.**

**Задание:**

****

**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

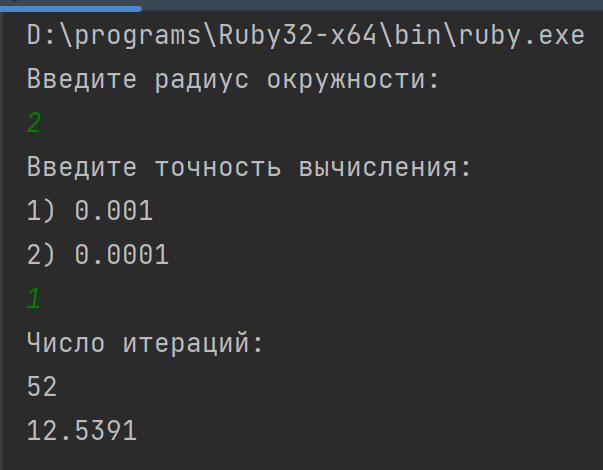
*Файл основной программы (main.rb):*

*# 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

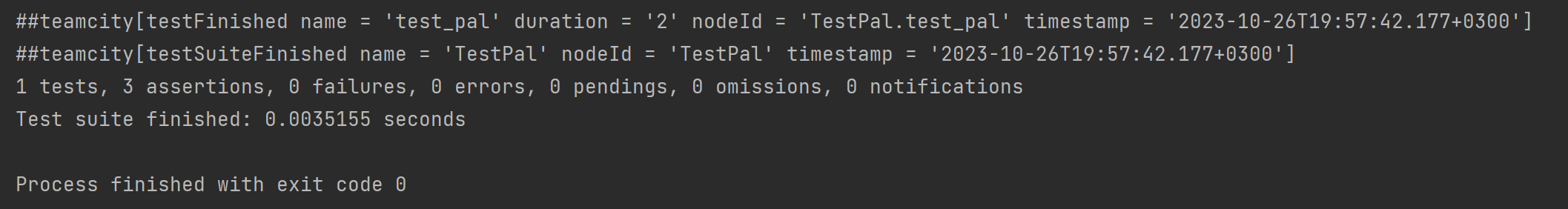
*Файл тестов (test.rb):*

*# 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

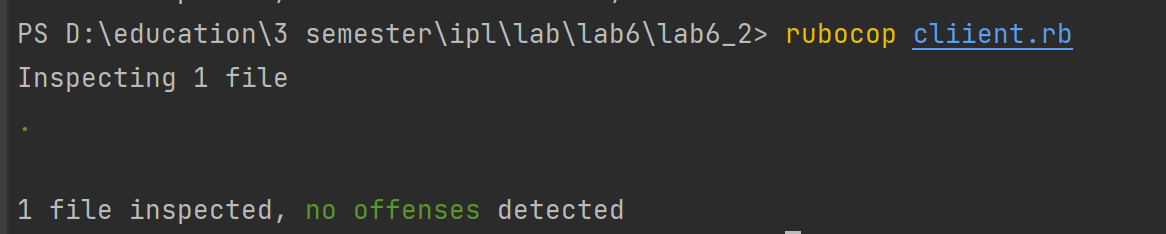
**Результат работы программы:**

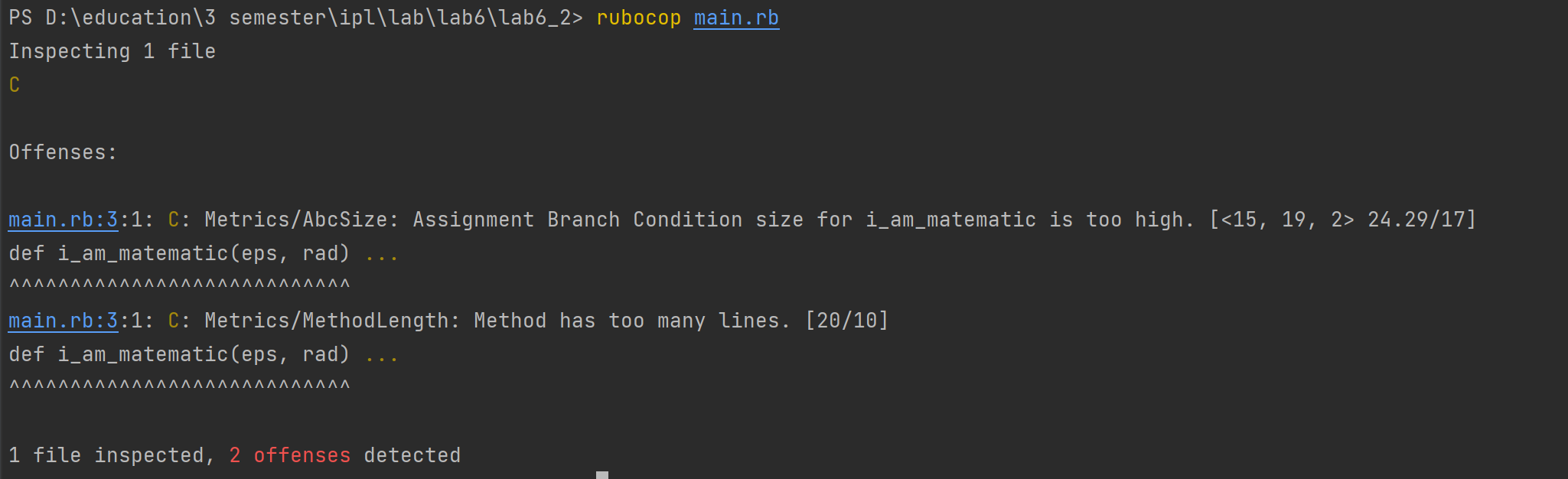
****

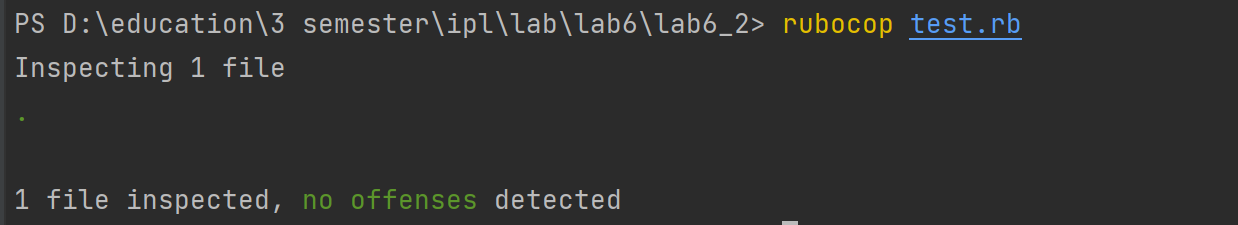
**Результат работы тестов:**

****

**Результат работы rubocop:**

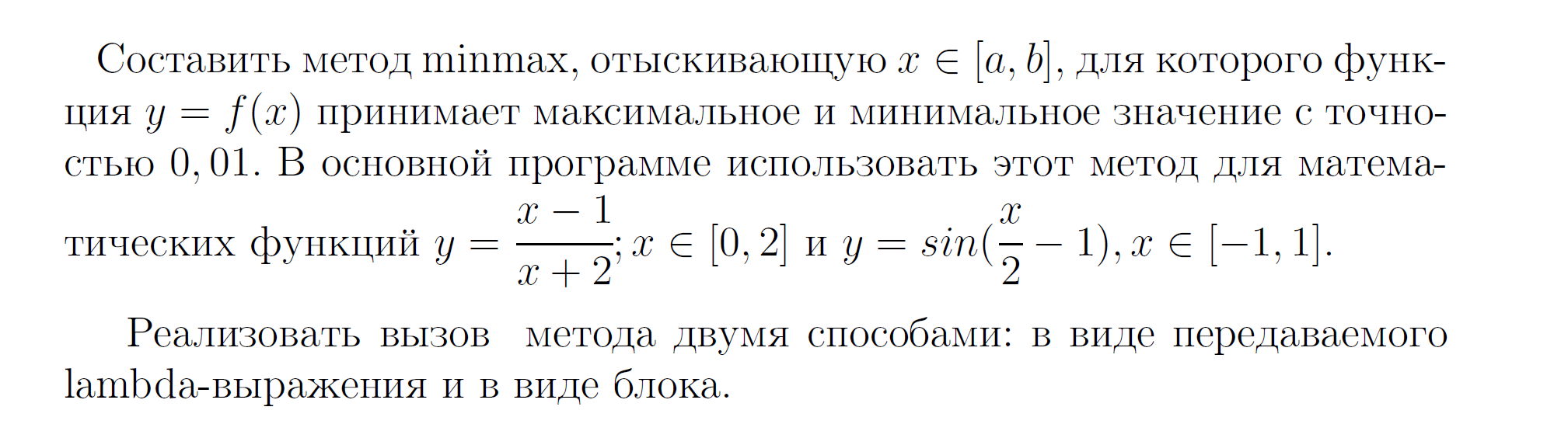
****

****

****

**Часть 3.**

**Задание:**

****

**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

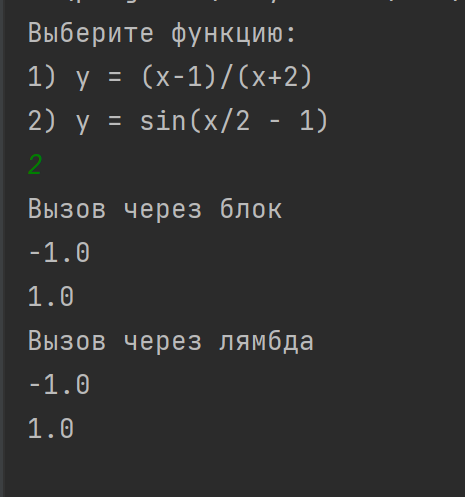
*Файл основной программы (main.rb):*

*# 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

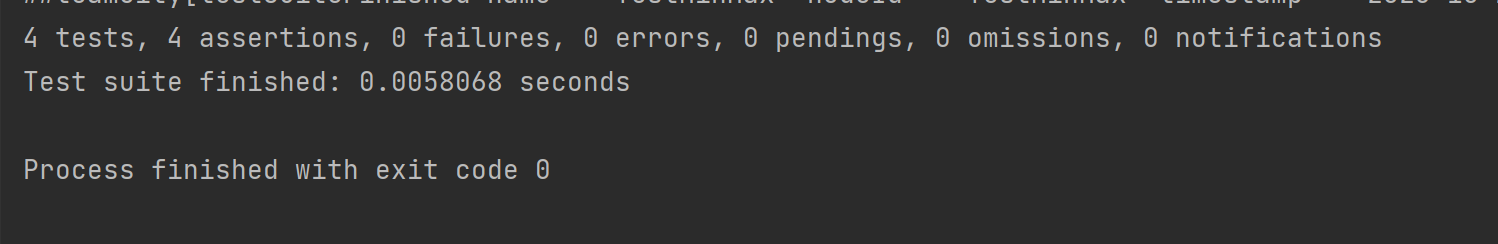
*Файл тестов (test.rb):*

*# 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

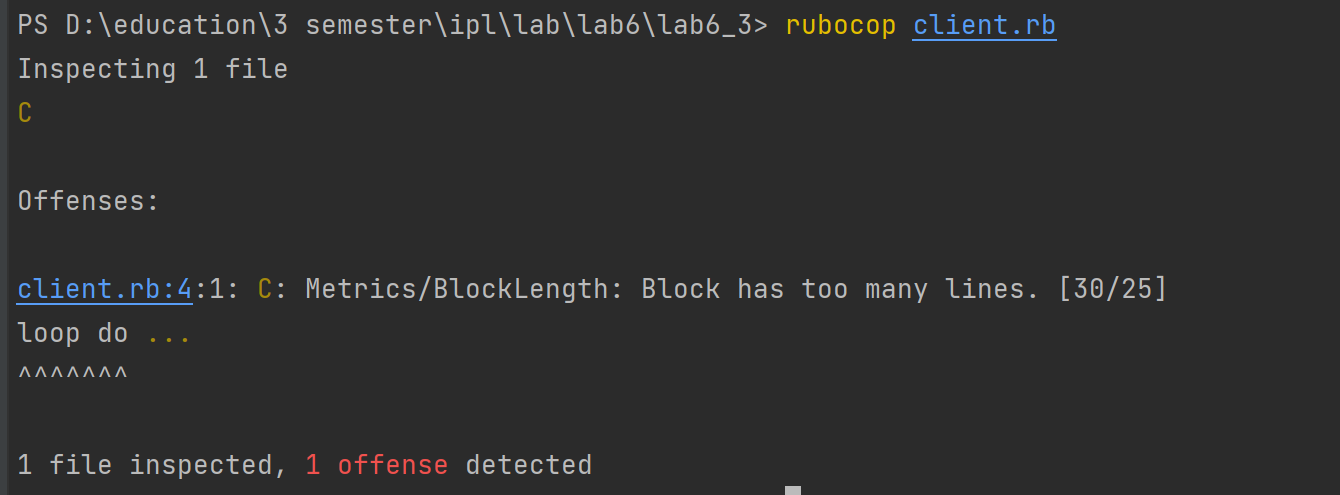
**Результат работы программы:**

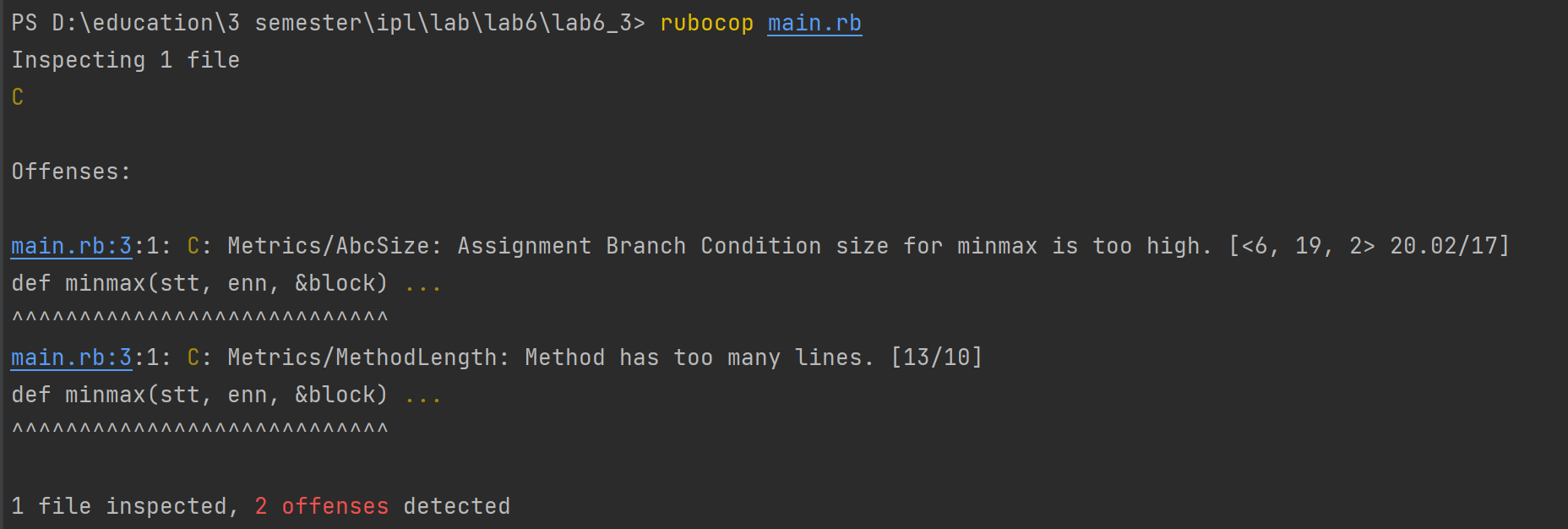
****

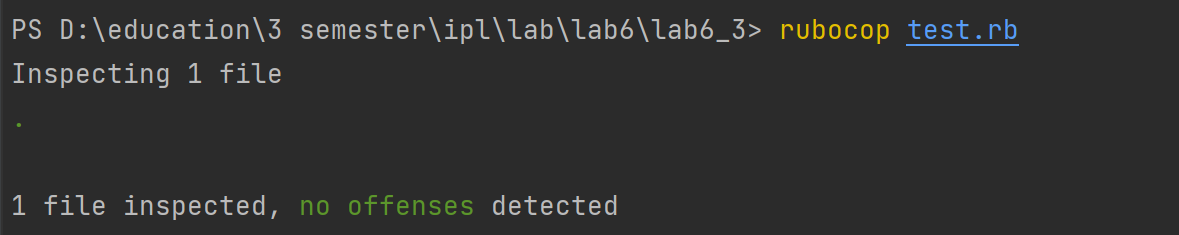
**Результат работы тестов:**

****

**Результат работы rubocop:**

****

****

****

**Вывод:** мы научились передавать в функцию блок программы, а также реализовывать функцию при помощи Enumerator или Enumerable