|  |  |
| --- | --- |
| 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 |

**Дисциплина:** Языки интернет-программирования

**Вариант:** 23

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-33Б |  |  | Д. И. Мироненко |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | В. Д. Шульман |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Часть 1**

**Задание**

A white background with black text

Description automatically generated

**Код программы**

**Файл “main.rb”**

|  |
| --- |
| # frozen\_string\_literal: true  require\_relative 'utils'  eps = 1e-4  puts "#{solve(eps)} (eps = #{eps})"  eps = 1e-5  puts "#{solve(eps)} (eps = #{eps})" |

**Файл “utils.rb”**

|  |
| --- |
| # frozen\_string\_literal: true  def \_factorial(num)  return 1 if num.zero?  result = 1  (1..num).each do |i|  result \*= i  end  result  end  def solve(eps)  expected = 1 / Math::E  sum = 1  k = 0  loop do  sum += ((-1)\*\*(k + 1)).to\_f / \_factorial(k + 1)  k += 1  break if (expected - sum).abs <= eps.to\_f  end  sum  end |

**Файл “test\_utils.rb”**

|  |
| --- |
| # frozen\_string\_literal: true  require 'minitest/autorun'  require\_relative 'utils'  # Test '\_factorial'  class TestFactorial < Minitest::Test  def test\_regular  assert\_equal 1, \_factorial(0)  assert\_equal 1, \_factorial(1)  assert\_equal 120, \_factorial(5)  end  end  # Test 'solve'  class TestSolve < Minitest::Test  def test\_regular  expected = 1 / Math::E  eps = 1e-4  assert\_in\_delta expected, solve(eps), eps  eps = 1e-5  assert\_in\_delta expected, solve(eps), eps  end  end |

**Работа программы**

A screenshot of a computer program

Description automatically generated

**Рисунок 1** – работа программы

**A screenshot of a computer program

Description automatically generated**

**Рисунок 2** – выполнение unit-тестов

**A screenshot of a computer

Description automatically generated**

**Рисунок 3** – проверка кода rubocop’ом

**Часть 2**

**Задание**



**Код программы**

**Файл “main.rb”**

|  |
| --- |
| # frozen\_string\_literal: true  require\_relative 'utils'  eps = 1e-4  puts "#{solve(eps)} (eps = #{eps})"  eps = 1e-5  puts "#{solve(eps)} (eps = #{eps})" |

**Файл “utils.rb”**

|  |
| --- |
| # frozen\_string\_literal: true  def \_factorial(num)  return 1 if num.zero?  (1..num).inject(:\*)  end  def solve(eps)  expected = 1 / Math::E  sequence = Enumerator.new do |x|  sum = 1  k = 0  loop do  sum += ((-1)\*\*(k + 1)).to\_f / \_factorial(k + 1)  x << sum  k += 1  end  end  sequence.find { |item| (expected - item).abs <= eps }  end |

**Файл “test\_utils.rb”**

|  |
| --- |
| # frozen\_string\_literal: true  require 'minitest/autorun'  require\_relative 'utils'  # Test '\_factorial'  class TestFactorial < Minitest::Test  def test\_regular  assert\_equal 1, \_factorial(0)  assert\_equal 1, \_factorial(1)  assert\_equal 120, \_factorial(5)  end  end  # Test 'solve'  class TestSolve < Minitest::Test  def test\_regular  expected = 1 / Math::E  eps = 1e-4  assert\_in\_delta expected, solve(eps), eps  eps = 1e-5  assert\_in\_delta expected, solve(eps), eps  end  end |

**Работа программы**

A screenshot of a computer

Description automatically generated

**Рисунок 4** – работа программы

**A screenshot of a computer program

Description automatically generated**

**Рисунок 5** – выполнение unit-тестов

**A screenshot of a computer screen

Description automatically generated**

**Рисунок 6** – проверка кода rubocop’ом

**Часть 3**

**Задание**

A math equations and formulas

Description automatically generated with medium confidence

**Код программы**

**Файл “main.rb”**

|  |
| --- |
| # frozen\_string\_literal: true  require\_relative 'utils'  points = [  [0, 1], [4, -0.76], [2, -0.42], [3, 0.41], [4, -0.65],  [5, 0.28], [4, -0.28], [6, -0.99], [5, -0.13], [9, -0.91]  ]  lambda\_f1 = ->(x) { Math.cos(x) }  lambda\_f2 = ->(x) { Math.sin(x\*\*2) }  puts 'Lambdas:'  points.each do |x, y|  result = neibr(x, y, &lambda\_f1)  puts "(#{x}, #{y}) belongs to y = cos(x)? - #{result}"  end  puts '-' \* 45  points.each do |x, y|  result = neibr(x, y, &lambda\_f2)  puts "(#{x}, #{y}) belongs to y = sin(x^2)? - #{result}"  end  puts "\n\nBlocks:"  points.each do |x, y|  result = neibr(x, y) { |value| block\_f1(value) }  puts "(#{x}, #{y}) belongs to y = cos(x)? - #{result}"  end  puts '-' \* 45  points.each do |x, y|  result = neibr(x, y) { |value| block\_f2(value) }  puts "(#{x}, #{y}) belongs to y = sin(x^2)? - #{result}"  end |

**Файл “utils.rb”**

|  |
| --- |
| # frozen\_string\_literal: true  EPS = 1e-2  def block\_f1(val)  Math.cos(val)  end  def block\_f2(val)  Math.sin(val\*\*2)  end  def neibr(x\_val, y\_val, &func)  (y\_val - func.call(x\_val)).abs <= EPS  end |

**Файл “test\_utils.rb”**

|  |
| --- |
| # frozen\_string\_literal: true  require 'minitest/autorun'  require\_relative 'utils'  # Test 'neibr'  class TestNeibr < Minitest::Test  def test\_f1\_with\_lambdas  points = [  [0, 1], [4, -0.76], [2, -0.42], [3, 0.41], [4, -0.65],  [5, 0.28], [4, -0.28], [6, -0.99], [5, -0.13], [9, -0.91]  ]  lambda\_f1 = ->(x) { Math.cos(x) }  expected = [  true, false, true, false, true,  true, false, false, false, true  ]  current\_result = []  points.each do |x, y|  current\_result << neibr(x, y, &lambda\_f1)  end  assert\_equal expected, current\_result  end  def test\_f2\_with\_lambdas  points = [  [0, 1], [4, -0.76], [2, -0.42], [3, 0.41], [4, -0.65],  [5, 0.28], [4, -0.28], [6, -0.99], [5, -0.13], [9, -0.91]  ]  lambda\_f2 = ->(x) { Math.sin(x\*\*2) }  expected = [  false, false, false, true, false,  false, true, true, true, false  ]  current\_result = []  points.each do |x, y|  current\_result << neibr(x, y, &lambda\_f2)  end  assert\_equal expected, current\_result  end  def test\_f1\_with\_blocks  points = [  [0, 1], [4, -0.76], [2, -0.42], [3, 0.41], [4, -0.65],  [5, 0.28], [4, -0.28], [6, -0.99], [5, -0.13], [9, -0.91]  ]  expected = [  true, false, true, false, true,  true, false, false, false, true  ]  current\_result = []  points.each do |x, y|  current\_result << neibr(x, y) { |value| block\_f1(value) }  end  assert\_equal expected, current\_result  end  def test\_f2\_with\_blocks  points = [  [0, 1], [4, -0.76], [2, -0.42], [3, 0.41], [4, -0.65],  [5, 0.28], [4, -0.28], [6, -0.99], [5, -0.13], [9, -0.91]  ]  expected = [  false, false, false, true, false,  false, true, true, true, false  ]  current\_result = []  points.each do |x, y|  current\_result << neibr(x, y) { |value| block\_f2(value) }  end  assert\_equal expected, current\_result  end  end |

**Работа программы**

A screenshot of a computer program

Description automatically generated

**Рисунок 7** – работа программы

**A screenshot of a computer program

Description automatically generated**

**Рисунок 8** – выполнение unit-тестов

**A computer screen shot of a program

Description automatically generated**

**Рисунок 9** – проверка кода rubocop’ом

**Вывод**

Реализованы несколько программ на языке программирования ruby; функции всех программ проверена на unit-тестах; исходный код проверен линтером rubocop на наличие ошибок.