Вариант 2.27.

Все консольные приложения Ruby следует реализовывать в виде трех отдельных файлов:

1. основная программа;
2. программа для взаимодействия с пользователем через консоль;
3. программа для автоматического тестирования на основе MiniTest::Unit. Везде, где это возможно, данные для проверки должны формироваться автоматически по правилам, указанным в задании.

Все тексты программ должны быть проверены на соответствие стилю программирования Ruby при помощи *rubocop* и *reek*.

ЛР 5

Часть 1

Вычислить: $a = x(\cos(z) + e^{-(x+3)})$.

Часть 2

С клавиатуры вводится целочисленный массив. Упорядочить массив по возврастанию и вывести сначала четные элементы, а затем нечетные. Выводить элементы с их индексами (порядковыми номерами) в исходном (несортированном) массиве в виде «индекс элемент, индекс элемент, ... ».

Автоматический тест программы обязательно должен генерировать случайные строки в соответствии с правилами, перечисленными в задании.

ЛР 6

Часть 1

Решить задачу, организовав итерационный цикл с точностью $\xi = 10^{-3}, 10^{-4}$. Вычислить длину кривой, определяемой функцией $y = ln\ x$ при $x \in [1, 2]$. Определить, как изменяется число разбиений при изменении точности.

Часть 2

Решить предыдущее задание с помощью Enumerator.

Часть 3

Составить метод intprg вычисления определенного интеграла по формуле прямоугольников: $S = \frac{b-a}{n} \sum_{i=1}^{n} f(x_i)$, где n – количество отрезков разбиения. В основной программе использовать метод intprg для вычисления интегралов: $\int_0^1 \frac{e^x}{x+1}\,\mathrm{d}x$ и $\int_0^2 x(x-1)\,\mathrm{d}x$.

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

ЛР 7

Часть 1

Организовать программным способом символьные файлы **F** и **G**. Определить совпадают ли компоненты этих файлов. Если нет, то получить номер первой компоненты, с которой начинаются различия. В случае, когда один из файлов имеет N компонент $N >= 0$ и повторяет начало другого (более длинного) файла, ответом должно быть число $N + 1$.

Автоматический тест программы обязательно должен проверять работу с файлами.

Часть 2

Разработать и реализовать иерархию классов для описанных объектов предметной области, используя механизмы наследования. Проверить ее на тестовом примере с демонстрацией всех возможностей разработанных классов на конкретных данных.

Объект, включающий поле — слово. Объект умеет выводить на экран значение своего поля и отвечать на запрос о его значении и количестве букв в слове.

Объект, включающий поля: целое число (длина слова) и слово. Объект умеет выводить на экран содержимое своих полей, возвращать по запросу их содержимое и количество согласных букв слова.

В тестирующей программе обеспечить автоматическую проверку того, что созданные объекты действительно соответствют заданной иерархии классов.

ЛР 8. Ruby on Rails

Разработать веб-приложение, имеющее HTML-страницу с формой ввода данных и HTML-страницу для представления результатов. Результат расчёта должен быть представлен в форме таблицы, оформленной с помощью элемента table или отдельными ячейками div и имеющей не менее двух колонок. Если по условию задания результат может быть представлен только в виде одной строки таблицы, необходимо реализовать вывод промежуточных результатов расчёта в качестве дополнительных строк. В этом случае первой колонкой таблицы будет порядковый номер итерации.

Под вводом с клавиатуры в тексте заданий следует понимать ввод в поле ввода данных формы на HTML-странице.

Текст задания:

Дано натуральное число m. Написать программу, определяющую такое натуральное число n, что двоичная запись числа n получается из двоичной записи числа m изменением порядка цифр на обратный порядок их следования. Например: $6 = 110$, а $3 = 011$. Вывести на печать числа и их двоичное представление.