Харківський національний університет ім. В.Н. Каразіна
Факультет комп’ютерних наук
Кафедра електроніки та управляючих систем

ЗВІТ
З КОНТРОЛЬНОЇ РОБОТИ №2
дисципліни: «Мови прикладного програмування»

Виконав: студентка групи ЗКС31
Москалюк С.Ю.

Перевірив:
Паршенцев Б.В.

Харків
2023

1. Як виконати обробку винятків (exception handling) в Ruby?
2. Що таке модуль в Ruby? Як ви створюєте та використовуєте модулі в Ruby?
3. Напишіть програму, яка приймає список чисел і сортує їх у порядку зростання або спадання за вибором користувача.
4. Напишіть програму, яка генерує випадковий пароль заданої довжини для користувача.

1. В Ruby, обробка винятків використовує блоки begin, rescue, та ensure. Ось приклад структури обробки винятків:

```ruby
begin
  # Код, в якому може виникнути виняток
  result = 10 / 0
rescue ZeroDivisionError => e
  # Обробка винятку ZeroDivisionError
  puts "Помилка ділення на нуль: #{e.message}"
rescue => e
  # Обробка будь-якого іншого винятку
  puts "Виникла помилка: #{e.message}"
else
  # Викликається, якщо винятку не виникло
  puts "Все виконано без помилок"
ensure
  # Викликається завжди, незалежно від того, чи був виняток
  puts "Завершується обробка винятків"
end
```

де begin: починає блок, в якому може виникнути виняток;

rescue: визначає блок, який виконується при виникненні конкретного винятку. Може бути кілька блоків rescue для обробки різних видів винятків. Також його можна використовувати без вказівки конкретного типу винятку, щоб «ловити» будь-які винятки;

else: викликається, якщо винятків в блоку begin не виявлено;

ensure: викликається завжди, незалежно від того, чи був виняток чи ні. Використовується для виконання завершальних операцій.

2. Модулі, які використовуються в Ruby, є інструментами для групування та організації коду, який містить методи, константи та інші функціональні елементи. Модулі дозволяють збирати код і використовувати його в інших програмах, класах або модулях.

Приклад створення та використання модулю:

```ruby
module MyModule
  MY_CONSTANT = 42

  def my_method
    puts "This is a method from MyModule"
```

```ruby
  end
end

class MyClass
  include MyModule

  def another_method
    puts "This is another method in MyClass"
  end
end

obj = MyClass.new
obj.my_method          # Виклик методу з модуля
puts MyModule::MY_CONSTANT  # Виклик константи з модуля
```

3.

```
D:\Ruby32-x64\bin\ruby.exe D:/practices/3year/ruby/kr2/kr2.rb
Enter your sequence divided by coma:1,8,4,23,5,7,2,3
 Select the sort order ('a' for ascending, 'd' for descending):d
 Sorted sequence: 23, 8, 7, 5, 4, 3, 2, 1

Process finished with exit code 0
```

Рисунок – 1 результат виконання завдання 3

Лістинг 1:

```ruby
def sort_numbers(numbers, order)
  sorted_numbers = numbers.sort

  if order.downcase == 'd'
    sorted_numbers.reverse!
  end

  sorted_numbers
end

print "Enter your sequence divided by coma: "
input = gets.chomp
numbers = input.split(',').map(&:to_i)

print "Select the sort order ('a' for ascending, 'd' for descending):
"
order = gets.chomp

sorted_numbers = sort_numbers(numbers, order)
puts "Sorted sequence: #{sorted_numbers.join(', ')}"
```

4.

```
D:\Ruby32-x64\bin\ruby.exe D:/practices/3year/ruby/kr2/kr2.2.rb
Enter password length:30
 Your generated password: tZrnV$jHtJFa!^7^v4QyEHPNgqxBSk

Process finished with exit code 0
```

Рисунок 2 – результат завдання 4

Лістинг 2:

```ruby
def generate_password(length)
  characters = ('a'..'z').to_a + ('A'..'Z').to_a + ('0'..'9').to_a +
['!', '@', '#', '$', '%', '^', '&', '*']

  password = Array.new(length) { characters.sample }
  password.join
end

print "Enter password length: "
password_length = gets.chomp.to_i

generated_password = generate_password(password_length)
puts "Your generated password: #{generated_password}"
```