Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра штучного інтелекту та програмного забезпечення

ЗВІТ
З КОНТРОЛЬНОЇ РОБОТИ №1
дисципліна: «Методи і технології розробки програмних систем»

Виконав: студент групи КС32
Синолицький Ілля

Харків
2024

Теоретичні питання

1. **Як у Ruby передаються аргументи в методи?**

   Аргументи передаються у методи за значенням, тобто передається копія змінної. Але якщо змінна містить об'єкт, передається посилання на об'єкт, тому зміни всередині методу можуть вплинути на сам об'єкт.

   Приклад:

```ruby
def newvalue(x)
    x += 10
    return x
end
y = 5
y = newvalue(y)
puts y  # Виведе 15

```

   Результат:

```
C:\Users\илья\OneDrive\Рабочий стол\ruby\KP1>ruby task1_1.rb
15

C:\Users\илья\OneDrive\Рабочий стол\ruby\KP1>
```

2. **Які є способи перетворення типів даних у Ruby?**

   У Ruby є 6 основних способів перетворення типів даних:

   - Перетворення в текст (String) — to_s
     555.to_s для перетворення числа на рядок
   - Перетворення в ціле число (Integer) — to_i
     "555".to_i рядок перетворюється на число 555
     "555a".to_i текст перетворюється на число 555, решта ігнорується
   - Перетворення в число з плаваючою крапкою (Float) — to_f
     "55.55".to_f текст перетворюється на число з крапкою
     "555a".to_f текст перетворюється на число 555.0 , текст ігноруємо
   - Перетворення в булеве значення (Boolean) — через !!
     !!555 будь-яке ненульове число стає true
     !!nil nil завжди false

- Перетворення в масив (Array) — to_a\
  (1..5).to_a діапазон перетворюється на масив [1, 2, 3, 4, 5]
- Перетворення в хеш (Hash) — to_h\
  [[:key1, 'value1'], [:key2, 'value2']].to_h масив перетворюється на хеш{:key1=>"value1", :key2=>"value2"}

3. Що таке винятки (exceptions) у Ruby та як їх обробляти?

Спосіб повідомити програміста про те, що сталася якась помилка під час виконання програми. Наприклад, якщо програма намагається поділити число на нуль або відкрити файл, якого не існує, це може спричинити збій. Щоб програма не зупинялась через такі помилки, Ruby дозволяє обробляти їх за допомогою спеціальних інструкцій.

Наприклад, ти відкриваєш двері. Якщо ключ підходить — все добре, ти заходиш (код виконується нормально). Якщо ключ не підходить (помилка), замість того, щоб просто стояти і нічого не робити (програма б зупинилась), ти пробуєш інший ключ або заходиш через інші двері (це обробка винятку).

```ruby
begin
  raise 'це виняток'
rescue RuntimeError => exception
  puts "Виняток оброблено: #{exception.inspect}"
  # => "Виняток оброблено: #<RuntimeError: це виняток>"
ensure
  puts "Цей код виконується завжди, незалежно від того, виникла помилка
end

puts "Це повідомлення *буде* виведене."

```

begin: Викликається виняток за допомогою raise, передаючи повідомлення 'це виняток'.

rescue: Ловиться виняток типу RuntimeError, і відбувається його обробка. Ми виводимо інформацію про виняток в консоль.

ensure: Цей блок виконується завжди, навіть якщо виняток не виникне або якщо він буде оброблений. Він корисний для того, щоб виконати якісь "очищення" (наприклад, закриття файлів, звільнення ресурсів).

puts "Це повідомлення *буде* виведене.": Після обробки винятка програма продовжує своє виконання, і це повідомлення виводиться, що підтверджує, що програма працює далі.

```
C:\Users\илья\OneDrive\Рабочий стол\ruby\КР1>ruby task1_3.rb
Виняток оброблено: #<RuntimeError: це виняток>
Цей код виконується завжди, незалежно від того, виникла помилка чи ні.
Це повідомлення *буде* виведене.
```

3. Як оголосити цикл while у Ruby?

```ruby
while умова
  # код який виконується поки умова є правильною
end
```

Практичні завдання

1. Напишіть програму, яка обчислює факторіал числа за допомогою рекурсії.

Реалізація:

```ruby
def factorial(x)
    return 1 if x == 0  # якщо n дорівнює 0, факторіал дорівнює 1
    x * factorial(x - 1) # Рекурсія x  факторіал числа (x-1)
  end

  puts factorial(5) # Виклик функції для числа 5

```

Рекзультати для 5!:

```
C:\Users\илья\OneDrive\Рабочий стол\ruby\КР1>ruby factorial.rb
120
```

Рекзультати для 10!:

```
C:\Users\илья\OneDrive\Рабочий стол\ruby\КР1>ruby factorial.rb
3628800
```

2. Створіть клас Employee, який містить ім'я, посаду та зарплату. Додайте метод, який збільшує зарплату на заданий відсоток

Реалізація:

```ruby
class Employee
    attr_accessor :name, :rang, :money

    def initialize(name, rang, money)
      @name = name
      @rang = rang
      @money = money
    end

    def increase_money(percentage)
      @money *= (1 + percentage / 100.0)
    end
  end

  # Отримання даних
  print "Введіть ім'я: "
  name = gets.chomp
  print "Введіть посаду: "
  rang = gets.chomp
  print "Введіть зарплату: "
  money = gets.chomp.to_f

  employee = Employee.new(name, rang, money)

  puts "Зарплата до підвищення: #{employee.money}"
  print "Введіть відсоток для підвищення зарплати: "
  employee.increase_money(gets.chomp.to_f)
  puts "Ім'я: #{employee.name}, Посада: #{employee.rang}"
  puts "Зарплата після підвищення: #{employee.money}"

```

Результати:

```
C:\Users\илья\OneDrive\Рабочий стол\ruby\КР1>ruby employer.rb
Введіть ім'я: Олег
Введіть посаду: Директор
Введіть зарплату: 100
Зарплата до підвищення: 100.0
Введіть відсоток для підвищення зарплати: 30
Ім'я: Олег, Посада: Директор
Зарплата після підвищення: 130.0
```