МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Кафедра програмних засобів

Звіт

з лабораторної роботи №8

з дисципліни: «Крос-платформне програмування»

за темою: «Багатонитевість при розробці програм на мові Ruby»

Виконав:

ст. гр. КНТ-117 М. І. Басанець

Прийняв:

старший викладач Є. М. Федорченко

2020

1 МЕТА РОБОТИ

Навчитись використовувати багатонитевість при розробці програм на мові Ruby.

2 зАВДАННЯ ДО РОБОТИ

1. Використайте програму, розроблену при виконанні лабораторної роботи № 2 (індивідуальне завдання). Модернізуйте програму так, щоб робота кожного об’єкту виконувалась в власному потоці. Реалізуйте роботи потоків таким чином, щоб перший запущений потік, завершився останнім, останній перший. Тобто всі потоки повинні завершитися в зворотному порядку щодо їх запуску.
2. Час для роботи одного потоку, має дорівнювати номеру варіанта. Також час між роботами потоків повинен дорівнювати значенню № варіанту / 2 (якщо варіант 5, то час між потоками повинен дорівнювати сумі часу роботи потоку + час між потоками = 7.5 с.). Вивести час виконання кожного потоку.
3. Реалізуйте обчислення суми мінімального елемента масиву і максимального. Кількість елементів масиву повинно рівнятися номеру варіанту. Обчислення мінімуму, максимуму та суми повинно проводитися в окремих потоках.

3 Результати виконання завдання

**Завдання 1:**

Код програми:

class Bird

def talk

return "I'm default bird. No one knows who I am"

end

def flying?

false

end

def name

return "DEFAULT BIRD"

end

end

class Chicken < Bird

def talk

return "I'm a chicken"

end

def flying?

return false

end

def name

return "Chicken"

end

end

class Swan < Bird

def talk

return "I'm a swan"

end

def flying?

return true

end

def name

return "Swan"

end

end

class Penguin < Bird

def talk

return "I'm a penguin"

end

def flying?

return false

end

def name

return "Penguin"

end

end

class Ostrich < Bird

def talk

return "I'm an ostrich"

end

def flying?

return false

end

def name

return "Ostrich"

end

end

chicken = Chicken.new()

swan = Swan.new()

penguin = Penguin.new()

ostrich = Ostrich.new()

thread1 = Thread.new {

puts("Thread #1 started")

thread2 = Thread.new {

puts("Thread #2 started")

thread3 = Thread.new {

puts("Thread #3 started")

thread4 = Thread.new {

puts("Thread #4 started")

puts ("#{ostrich.name} says: #{ostrich.talk}. Is flying: #{ostrich.flying?}")

puts("Thread #4 finished")

}

thread4.join()

puts ("#{penguin.name} says: #{penguin.talk}. Is flying: #{penguin.flying?}")

puts("Thread #3 finished")

}

thread3.join()

puts ("#{swan.name} says: #{swan.talk}. Is flying: #{swan.flying?}")

puts("Thread #2 finished")

}

thread2.join()

puts ("#{chicken.name} says: #{chicken.talk}. Is flying: #{chicken.flying?}")

puts("Thread #1 finished")

}

thread1.join()

Результати роботи програми зображено на рисунку 3.1.

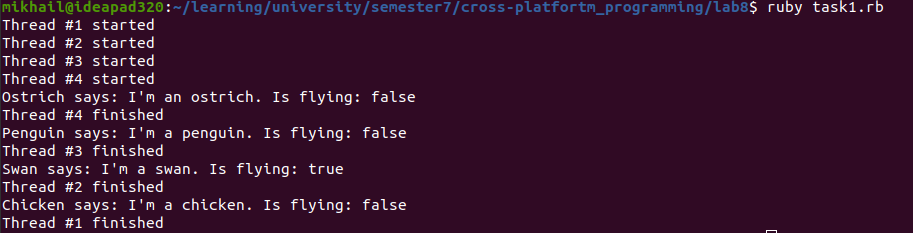


Рисунок 3.1 – Результати роботи програми 1

**Завдання 2:**

Код програми:

thread\_working\_time = 2

thread1 = Thread.new {

puts ("Thread 1 started at #{Time.now}")

sleep(thread\_working\_time)

puts ("Thread 1 finished at #{Time.now}")

}

thread1.join()

sleep(thread\_working\_time / 2)

thread2 = Thread.new {

puts ("Thread 2 started at #{Time.now}")

sleep(thread\_working\_time)

puts ("Thread 2 finished at #{Time.now}")

}

thread2.join()

Результати роботи програми зображено на рисунку 3.2.

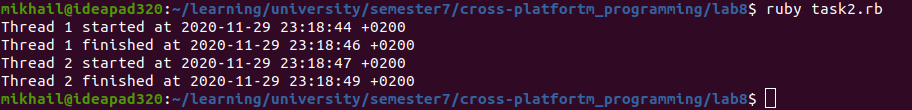


Рисунок 3.2 – Результати роботи програми 2

**Завдання 3:**

Код програми:

arr = Array.new(2){rand(0..100)}

thread\_max = Thread.new{Thread.current[:output] = arr.max()}

thread\_min = Thread.new{Thread.current[:output] = arr.min()}

puts arr

thread\_max.join()

thread\_min.join()

max = thread\_max[:output]

min = thread\_min[:output]

puts("Max element is #{max}, min element is #{min}")

puts("Sum of max and min elements is #{min + max}")

Результати роботи програми зображено на рисунку 3.3.

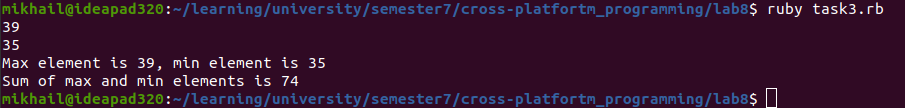


Рисунок 3.3 – Результати роботи програми 3

4 висновки

При виконанні даної лабораторної роботи на практиці засвоїв основні принципи та засади роботи з багатонитевістю у Ruby.

Відповіді на контрольні запитання:

1. Що таке багатонитевість?

Багатонитевість - принцип виконання програми, при якому декілька процесів можуть виконуватися одночасно, не заважаючи один одному.

2. Як виконується багатонитевість за допомогою Ruby?

В Ruby потоки не залежать від операційної системи. Для створення потоку використовується клас Thread.

3. Які існують методи для роботи з потоками в Ruby?

Основні методи роботи з потоками (методи класу Thread): new (створення нового потоку), current (повертає потік, що виконується), pass (запускає планувальник потоків, щоб передати виконання іншій потік), join (призупиняє основний потік та запускає новий), kill (завершує потік), stop (зупиняє виконання поточного потоку, поміщаючи його в сплячий стан), start (запускає сплячий потік), exit (вихід з потоку).

4. Які існують стани потоків в Ruby?

sleep, run, aborting, false (потік завершився нормально), nil (помикла при виконанні потоку). Отримати значення потоку можна за допомогою методу status().

5. Можливо використовувати глобальні змінні основного потоку в нових потоках?

Можливо, але необхідно забезпечити контроль доступу потоків до даних.