

Universidad nacional Autónoma de Nicaragua Unan-León



Ingeniería en Telemática

Componente: Software como servicio.

Nombres: Ingrid Valeria Ruiz Ulloa 21-00483-0

Lenuel Gastón Pereira Hernández 19-03199-0

Nombre del Docente: Ervin Montes

“A la libertad de la universidad”

1 y 2 Array y método each

```
lenuelp@Debian12: ~/Desktop/ruby2
lenuelp@Debian12:~/Desktop/ruby2$ ruby2 array.rb
bash: ruby2: command not found
lenuelp@Debian12:~/Desktop/ruby2$ ruby array.rb
lunes
martes
miércoles
jueves
viernes
sábado
domingo

Imprimir por posición
martes
jueves
sábado
lenuelp@Debian12:~/Desktop/ruby2$ ruby Méthodoeach.rb
dia 0=lunes
dia 1=martes
dia 2=miércoles
dia 3=jueves
dia 4=viernes
dia 5=sábado
dia 6=domingo
lenuelp@Debian12:~/Desktop/ruby2$
```

. **El método each** en Ruby se utiliza como iterador para recorrer un array, tomando como ejemplo el programa anterior, crear uno nuevo y utilizar el método each para recorrer el array e imprimirlo por pantalla.

3. **Métodos para trabajar con array** En Ruby existen muchos métodos específicamente para trabajar con array, entre los cuales se pueden encontrar: **pop, push, join, last, split**. En este enunciado se mostrará el funcionamiento de algunos de ellos, los cuales son muy útiles en el desarrollo de aplicaciones en donde se trabaja con el lenguaje Ruby

```

lenuelp@Debian12:~/Desktop/ruby2$ ruby 3.rb
Array en Ruby
lunes
martes
miércoles
jueves
viernes
sábado
domingo

Método to_s
["lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo"]

Método join
lunes,martes,miércoles,jueves,viernes,sábado,domingo

Método first
lunes

Método last
domingo

Método length
7

```

3.3 Modificar el programa anterior, y hacer uso de los métodos push y pop para ver la diferencia del comportamiento entre ambos, en relación a su uso sobre lo

```

lenuelp@Debian12:~/Desktop/ruby2$ ruby 3.3.rb
Array completo
["lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo"]

Método POP
domingo

Método length
6

Último dato
sábado

Método PUSH
lunes
martes
miércoles
jueves
viernes
sábado
final

Tamaño nuevo
7

```

```

lenuelp@Debian12:~/Desktop/ruby2$ ruby 3.3.rb
Array completo
["lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo"]

Método POP
domingo

Método length
6

Último dato
sábado

Método PUSH
lunes
martes
miércoles
jueves
viernes
sábado
final

Tamaño nuevo
7

```

Recorte y anote

Recorte guardado

Seleccione uno de los arrays.

4. Métodos propios

En Ruby como en cualquier otro lenguaje de programación se pueden definir métodos para que realicen cierto trabajo, para entender un poco mejor de esto, crear un nuevo programa llamado `metodos_propios.rb` y agregar el siguiente código.

```
lenuelp@Debian12:~/Desktop/ruby2$ ruby metodopropio.rb
Ingrese su nombre
lenu
Bienvenido LENU
Ingrese su año de nacimiento
2001
Ingrese el año actual
2024

Tu edad actual es 23 años
lenuelp@Debian12:~/Desktop/ruby2$
```

5. Hash

Algo muy utilizado en el lenguaje Ruby son los hashes al momento de trabajar con datos. Crear un programa llamado `hash.rb` y agregar el código a continuación

```
lenuelp@Debian12:~/Desktop/ruby2$ ruby hash.rb
rojo  : #FF0000
verde : #008000
azul  : #0000FF
lenuelp@Debian12:~/Desktop/ruby2$
```

5.2 Para ver de otra manera el funcionamiento, crear un nuevo programa `hash_2.rb` y agregar el código.

```

lenuelp@Debian12: ~/Desktop/ruby2$ ruby hash2.rb
Nombre de usuario: Juan Pérez

Correo: JuanP@example.com
lenuelp@Debian12: ~/Desktop/ruby2$ █

```

6. Clases

Crear un programa clases.rb, en el cual se creará una clase Palíndromo que contendrá un método para verificar una frase ingresada, como aparece a continuación

The screenshot shows a terminal window on the right and a Visual Studio Code editor on the left. The terminal window displays the execution of the 'clases.rb' script, which prompts the user to enter a phrase. The user enters 'oro', and the script outputs 'La frase oro Es palindromo'. The Visual Studio Code editor shows the source code of 'clases.rb', which defines a 'Palindromo' class with a 'vefificar_frase' method. The method checks if the input string is equal to its reverse. If it is, it prints 'La frase #{frase} Es palindromo'; otherwise, it prints 'La frase #{frase} No es palindromo'. The script also includes a main section that prompts the user for a phrase, reads it, creates a new 'Palindromo' object, and calls the 'vefificar_frase' method.

```

... 3.3.rb  metodopropio.rb  hash.rb  hash2.rb  lenuelp@Debian12: ~/Desktop/ruby2$ ruby clases.rb
Ingrese una frase
oro
lenuelp@Debian12: ~/Desktop/ruby2$ ruby clases.rb
Ingrese una frase
oro
lenuelp@Debian12: ~/Desktop/ruby2$ ruby clases.rb
Ingrese una frase
oro
La frase oro Es palindromo
lenuelp@Debian12: ~/Desktop/ruby2$ █

clases.rb - ruby2 - Visual Studio Code
lenuelp@Debian12: ~/Desktop/ruby2$
1 class Palindromo
2
3   def vefificar_frase(frase)
4     if frase == frase.reverse
5       puts "La frase #{frase} Es palindromo"
6     else
7       puts "La frase #{frase} No es palindromo"
8     end
9   end
10 end
11 end
12 puts "Ingrese una frase"
13 frase = gets.chomp
14 verificador = Palindromo.new
15 verificador.vefificar_frase(frase)

```

Una frase Palíndromo es una palabra o expresión, que es igual si se lee de izquierda a derecha que de derecha a izquierda. Puede ejecutar el programa de nuevo e interactuar con el funcionamiento, para realizar distintas pruebas ingresando otras palabras o frases. Ejemplo: “anitalavalatina”

7. Variable de instancia.

Las variables de instancia son variables de un objeto, una de las diferencias de las variables locales es que estas existen hasta que el método ha terminado e inician con arroba “@”.

Crear un programa en Ruby llamado variables.rb y escribir lo siguiente:

```
lenuelp@Debian12:~/Desktop/ruby2$ ruby variable.rb
cuantas veces desea lanzar el dado
5
```

Lanzamiento

6

Lanzamiento

4

Lanzamiento

3

Lanzamiento

1

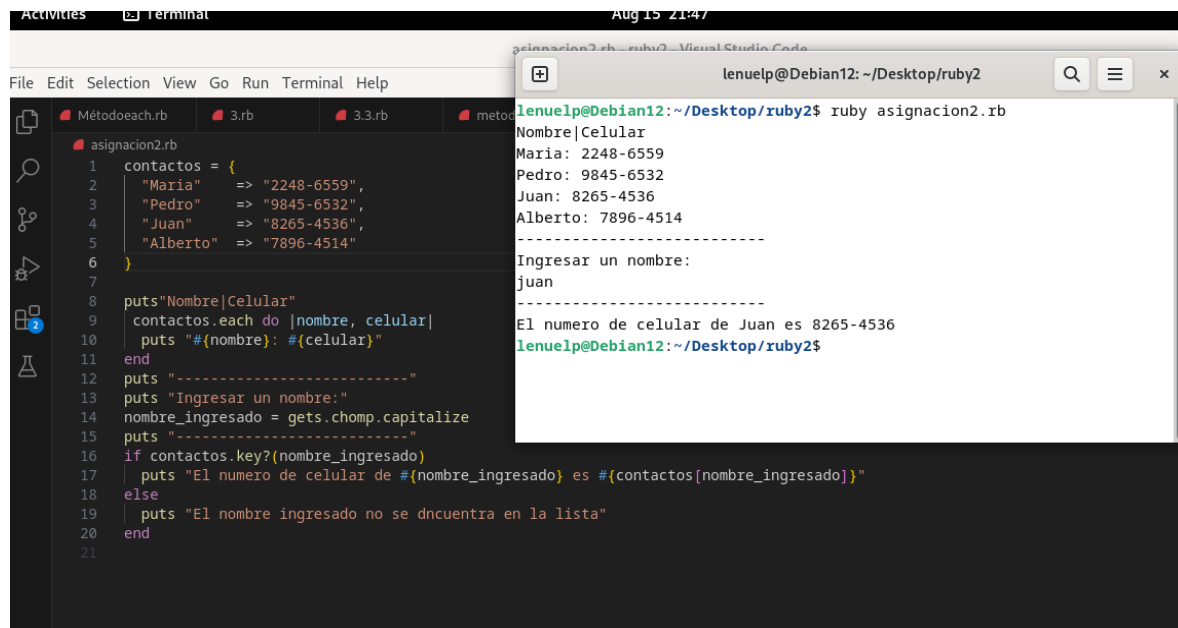
Lanzamiento

6

-

Ejercicios propuestos para ser entregados al docente

Crear un programa en Ruby que contenga un hash, el cual este compuesto de nombre = clave y celular = valor, el programa deberá mostrar el hash completo, solicitar el nombre que sería la clave y retornar el celular que sería el valor, correspondiente a ese nombre. Deberá validar si el dato existe en el hash y que cuando se ingrese un nombre en minúscula a como se muestra en la figura 32, el nombre Juan se ingresó en minúscula y el programa devuelve el celular correspondiente al nombre



```
lenuelp@Debian12:~/Desktop/ruby2$ ruby asignacion2.rb
Nombre|Celular
Maria: 2248-6559
Pedro: 9845-6532
Juan: 8265-4536
Alberto: 7896-4514
-----
Ingresar un nombre:
juan
-----
El numero de celular de Juan es 8265-4536
lenuelp@Debian12:~/Desktop/ruby2$
```

```
1 contactos = {
2   "Maria" => "2248-6559",
3   "Pedro" => "9845-6532",
4   "Juan"  => "8265-4536",
5   "Alberto" => "7896-4514"
6 }
7
8 puts "Nombre|Celular"
9 contactos.each do |nombre, celular|
10  puts "#{nombre}: #{celular}"
11 end
12 puts "-----"
13 puts "Ingresar un nombre:"
14 nombre_ingresado = gets.chomp.capitalize
15 puts "-----"
16 if contactos.key?(nombre_ingresado)
17   puts "El numero de celular de #{nombre_ingresado} es #{contactos[nombre_ingresado]}"
18 else
19   puts "El nombre ingresado no se encuentra en la lista"
20 end
21
```

Realice un programa en Ruby que solicite por pantalla un número cualquiera y que imprima la suma de los números pares e impares que componen el número ingresado, para la solución crear una clase de nombre Calcular la cual contendrá 2 métodos, el primer método para los cálculos de los números pares y el segundo método para los cálculos de los número impares, se deberá mostrar a como se muestra

```
PHP: 1.49 suma_pares_impares.rb X Extension: Ruby
suma_pares_impares.rb
1 class Calcular
2   # Método para calcular la suma de los números pares
3   def suma_pares(numero)
4     suma = 0
5     numero.to_s.each_char do |digito|
6       suma += digito.to_i if digito.to_i.even?
7     end
8     suma
9   end
10
11   # Método para calcular la suma de los números impares
12   def suma_impares(numero)
13     suma = 0
14     numero.to_s.each_char do |digito|
15       suma += digito.to_i if digito.to_i.odd?
16     end
17     suma
18   end
19 end
20
21 # Solicitar un número al usuario
22 puts "Ingrese un número:"
23 numero = gets.chomp.to_i
24
25 # Crear una instancia de la clase Calcular
26 calcular = Calcular.new
27
28 # Calcular y mostrar las sumas
29 suma_pares = calcular.suma_pares(numero)
30 suma_impares = calcular.suma_impares(numero)
31
32 puts "La suma de los números pares es: #{suma_pares}"
33 puts "La suma de los números impares es: #{suma_impares}"
```