

**UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA  
UNAN-León.**



**Facultad de Ciencias y Tecnología**

**Carrera:** Ingeniería en Telemática.

**Componente:** Software como un Servicio.

**Docente:** Erving Montes.

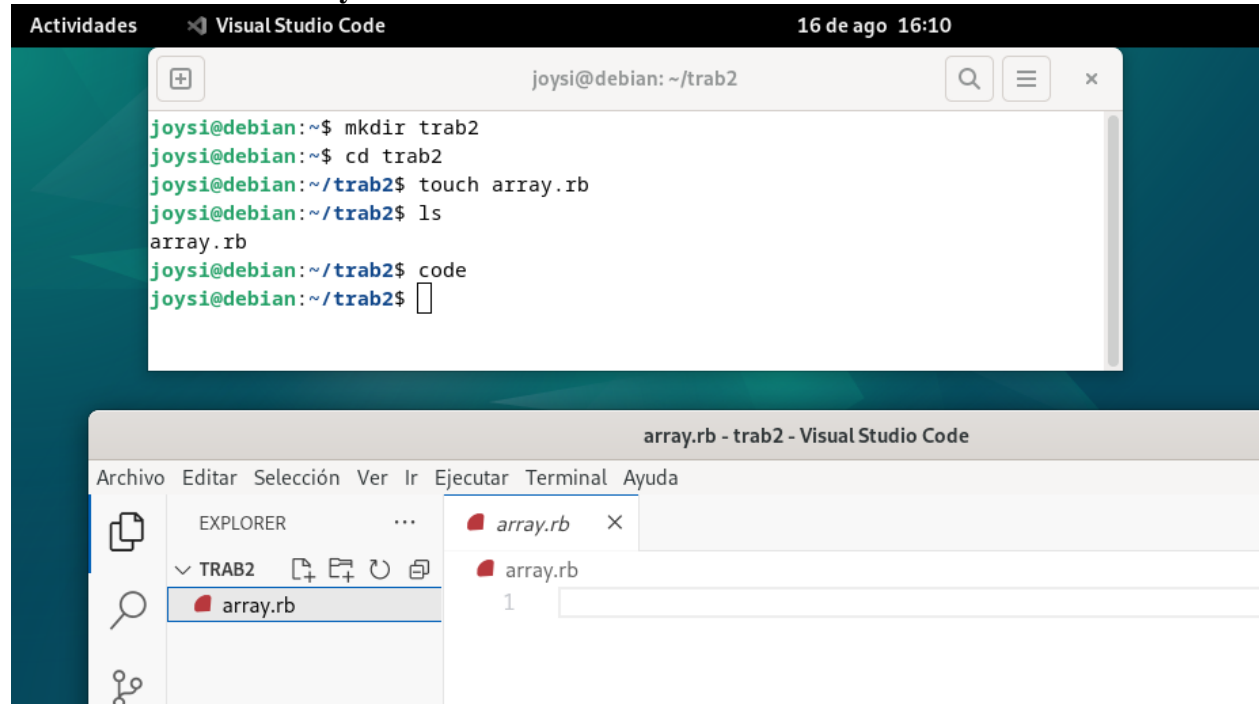
**Nombre:**

- Joysi Julissa García Soza.

**Fecha:** 16/08/2024.

**¡A la libertad por la universidad!**

## Creacion de directorios y demás.

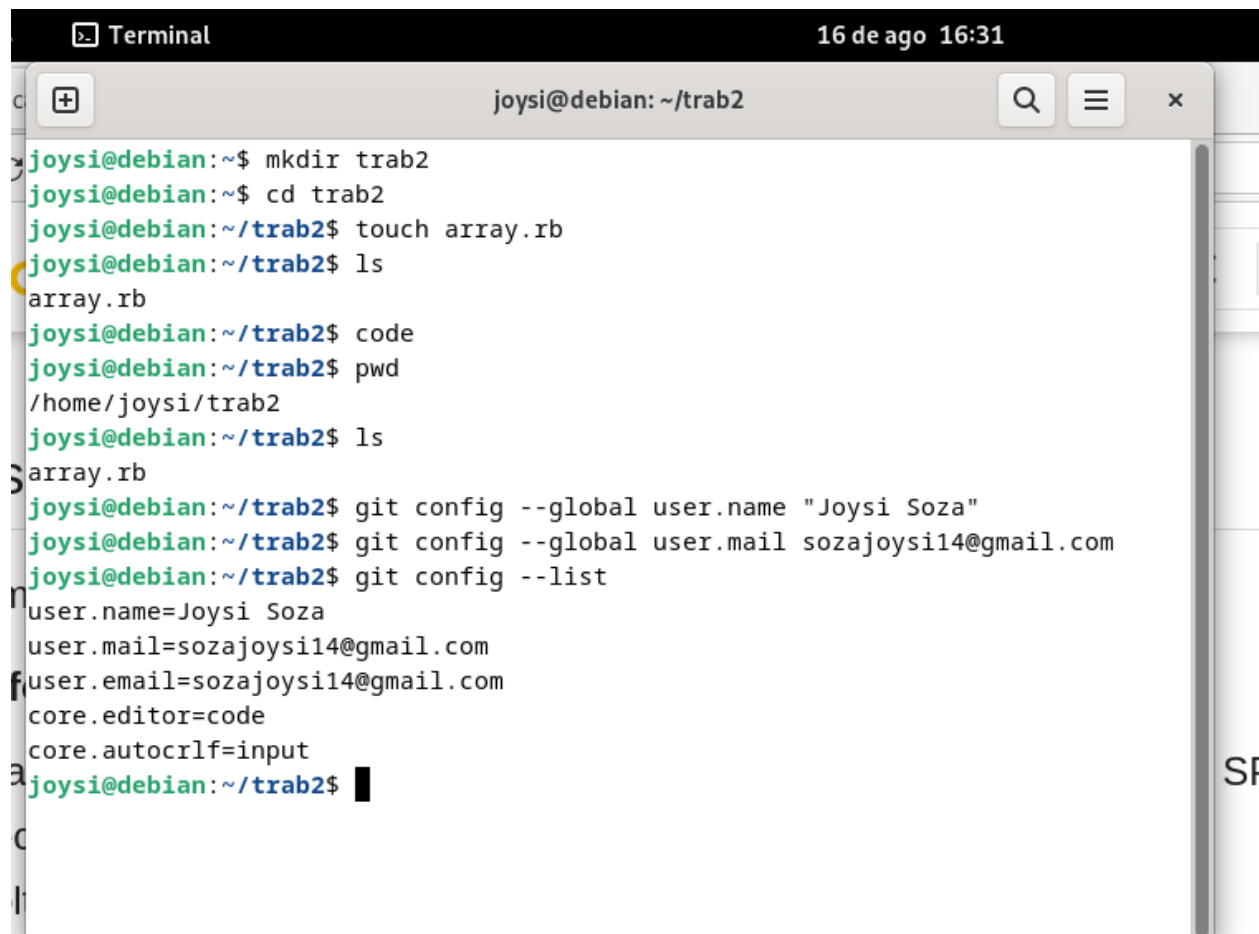


The screenshot shows the Visual Studio Code interface. At the top, the title bar reads "Visual Studio Code" and "16 de ago 16:10". The main window has a terminal pane with the following commands and output:

```
joysi@debian:~$ mkdir trab2
joysi@debian:~$ cd trab2
joysi@debian:~/trab2$ touch array.rb
joysi@debian:~/trab2$ ls
array.rb
joysi@debian:~/trab2$ code
joysi@debian:~/trab2$
```

Below the terminal, the Explorer sidebar shows a folder named "TRAB2" containing a file named "array.rb". The main editor area shows the "array.rb" file with a single line of code:

```
1
```



The screenshot shows a terminal window titled "Terminal" with the date and time "16 de ago 16:31". The terminal shows the same initial commands as the previous screenshot, followed by git configuration commands:

```
joysi@debian:~$ mkdir trab2
joysi@debian:~$ cd trab2
joysi@debian:~/trab2$ touch array.rb
joysi@debian:~/trab2$ ls
array.rb
joysi@debian:~/trab2$ code
joysi@debian:~/trab2$ pwd
/home/joysi/trab2
joysi@debian:~/trab2$ ls
array.rb
joysi@debian:~/trab2$ git config --global user.name "Joysi Soza"
joysi@debian:~/trab2$ git config --global user.mail sozajoysi14@gmail.com
joysi@debian:~/trab2$ git config --list
user.name=Joysi Soza
user.mail=sozajoysi14@gmail.com
user.email=sozajoysi14@gmail.com
core.editor=code
core.autocrlf=input
joysi@debian:~/trab2$
```

```
ades Terminal 16 de ago 16:37
joysi@debian: ~/trab2
joysi@debian:~/trab2$ ls
array.rb
joysi@debian:~/trab2$ touch each.rb
joysi@debian:~/trab2$ la
bash: la: orden no encontrada
joysi@debian:~/trab2$ ls
array.rb  each.rb
joysi@debian:~/trab2$ touch ejer3.rb ejer4.rb ejer5.rb ejer6.rb ejer.7
joysi@debian:~/trab2$ ls
array.rb  each.rb  ejer3.rb  ejer4.rb  ejer5.rb  ejer6.rb  ejer.7
joysi@debian:~/trab2$
```

te aparecerá este símbolo "\"

Git:

SP,

```
ades Terminal 16 de ago 16:38
joysi@debian: ~/trab2
array.rb each.rb
joysi@debian:~/trab2$ touch ejer3.rb ejer4.rb ejer5.rb ejer6.rb ejer.7
joysi@debian:~/trab2$ ls
array.rb each.rb ejer3.rb ejer4.rb ejer5.rb ejer6.rb ejer.7
joysi@debian:~/trab2$ ls -a
. .. array.rb each.rb ejer3.rb ejer4.rb ejer5.rb ejer6.rb ejer.7
joysi@debian:~/trab2$ git init
ayuda: Usando 'master' como el nombre de la rama inicial. Este nombre de rama
predeterminado
ayuda: está sujeto a cambios. Para configurar el nombre de la rama inicial par
a usar en todos
ayuda: de sus nuevos repositorios, reprimiendo esta advertencia, llama a:
ayuda:
ayuda: git config --global init.defaultBranch <nombre>
ayuda:
ayuda: Los nombres comúnmente elegidos en lugar de 'master' son 'main', 'trunk
' y
ayuda: 'development'. Se puede cambiar el nombre de la rama recién creada medi
ante este comando:
ayuda:
ayuda: git branch -m <nombre>
Inicializado repositorio Git vacío en /home/joysi/trab2/.git/
joysi@debian:~/trab2$
```

```
des Terminal 16 de ago 16:39
joysi@debian: ~/trab2
joysi@debian:~/trab2$ ls -a
. array.rb ejer3.rb ejer5.rb ejer.7
.. each.rb ejer4.rb ejer6.rb .git
joysi@debian:~/trab2$
```

```
des Terminal 16 de ago 16:41
plec joysi@debian: ~/trab2
joysi@debian:~/trab2$ ls -a
. array.rb ejer3.rb ejer5.rb ejer.7
.. each.rb ejer4.rb ejer6.rb .git
joysi@debian:~/trab2$ touch ejercicio1real.rb
joysi@debian:~/trab2$ touch ejercicio2real.rb
joysi@debian:~/trab2$ ls
array.rb ejer3.rb ejer5.rb ejer.7 ejercicio2real.rb
each.rb ejer4.rb ejer6.rb ejercicio1real.rb
joysi@debian:~/trab2$ ls -a
. array.rb ejer3.rb ejer5.rb ejer.7 ejercicio2real.rb
.. each.rb ejer4.rb ejer6.rb ejercicio1real.rb .git
joysi@debian:~/trab2$
```

```
dades Terminal 16 de ago 16:42
plec joysi@debian: ~/trab2
joysi@debian:~/trab2$ ls
array.rb ejer3.rb ejer5.rb ejer7.rb ejercicio2real.rb
each.rb ejer4.rb ejer6.rb ejercicio1real.rb
joysi@debian:~/trab2$
```

```
ades Terminal 16 de ago 16:43
joysi@debian: ~/trab2
joysi@debian:~/trab2$ ls
array.rb  ejer3.rb  ejer5.rb  ejer7.rb  ejercicio2real.rb
each.rb   ejer4.rb  ejer6.rb  ejercicio1real.rb
joysi@debian:~/trab2$ git status
En la rama master

No hay commits todavía

Archivos sin seguimiento:
(usa "git add <archivo>..." para incluirlo a lo que será confirmado)
    array.rb
    each.rb
    ejer3.rb
    ejer4.rb
    ejer5.rb
    ejer6.rb
    ejer7.rb
    ejercicio1real.rb
    ejercicio2real.rb

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa
"git add" para hacerles seguimiento)
joysi@debian:~/trab2$
```

```
s Terminal 16 de ago 16:45
ec joysi@debian: ~/trab2
joysi@debian:~/trab2$ ls -a
.   array.rb  ejer3.rb  ejer5.rb  ejer7.rb  ejercicio2real.rb
..  each.rb   ejer4.rb  ejer6.rb  ejercicio1real.rb  .git
joysi@debian:~/trab2$
```

```
Terminal 16 de ago 16:45
joysi@debian: ~/trab2
joysi@debian:~/trab2$ git status
En la rama master

No hay commits todavía

Archivos sin seguimiento:
(usa "git add <archivo>..." para incluirlo a lo que será confirmado)
    array.rb
    each.rb
    ejer3.rb
    ejer4.rb
    ejer5.rb
    ejer6.rb
    ejer7.rb
    ejercicio1real.rb
    ejercicio2real.rb

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para hacerles seguimiento)
joysi@debian:~/trab2$
```

```
Terminal 16 de ago 16:48
joysi@debian: ~/trab2
joysi@debian:~/trab2$ git status
En la rama master

No hay commits todavía

Archivos sin seguimiento:
(usa "git add <archivo>..." para incluirlo a lo que será confirmado)
    array.rb
    each.rb
    ejer3.rb
    ejer4.rb
    ejer5.rb
    ejer6.rb
    ejer7.rb
    ejercicio1real.rb
    ejercicio2real.rb

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para hacerles seguimiento)
joysi@debian:~/trab2$ git add array.rb each.rb ejer3.rb ejer4.rb ejer5.rb ejer6.rb ejer7.rb ejercicio1real.rb ejercicio2real.rb
joysi@debian:~/trab2$
```

```
Terminal 16 de ago 16:49
joysi@debian: ~/trab2
joysi@debian:~/trab2$ git status
En la rama master

No hay commits todavía

Cambios a ser confirmados:
(usa "git rm --cached <archivo>..." para sacar del área de stage)
nuevos archivos: array.rb
nuevos archivos: each.rb
nuevos archivos: ejer3.rb
nuevos archivos: ejer4.rb
nuevos archivos: ejer5.rb
nuevos archivos: ejer6.rb
nuevos archivos: ejer7.rb
nuevos archivos: ejercicio1real.rb
nuevos archivos: ejercicio2real.rb

joysi@debian:~/trab2$
```

```
Terminal 16 de ago 16:56
joysi@debian: ~/trab2
joysi@debian:~/trab2$ git add *.rb
joysi@debian:~/trab2$ git diff *.rb
joysi@debian:~/trab2$ ls
array.rb  ejer3.rb  ejer5.rb  ejer7.rb  ejercicio2real.rb
each.rb  ejer4.rb  ejer6.rb  ejercicio1real.rb
joysi@debian:~/trab2$ git status
En la rama master

No hay commits todavía

Cambios a ser confirmados:
(usa "git rm --cached <archivo>..." para sacar del área de stage)
nuevos archivos: array.rb
nuevos archivos: each.rb
nuevos archivos: ejer3.rb
nuevos archivos: ejer4.rb
nuevos archivos: ejer5.rb
nuevos archivos: ejer6.rb
nuevos archivos: ejer7.rb
nuevos archivos: ejercicio1real.rb
nuevos archivos: ejercicio2real.rb

joysi@debian:~/trab2$
```



```
Terminal 16 de ago 16:57

joysi@debian: ~/trab2

Cambios a ser confirmados:
(usa "git rm --cached <archivo>..." para sacar del área de stage)
nuevos archivos: array.rb
nuevos archivos: each.rb
nuevos archivos: ejer3.rb
nuevos archivos: ejer4.rb
nuevos archivos: ejer5.rb
nuevos archivos: ejer6.rb
nuevos archivos: ejer7.rb
nuevos archivos: ejercicio1real.rb
nuevos archivos: ejercicio2real.rb

joysi@debian:~/trab2$ git commit -m "creacion de 7 ejemplos y 2 ejercicios"
[master (commit-raiz) 4ab4dcf] creacion de 7 ejemplos y 2 ejercicios
9 files changed, 7 insertions(+)
create mode 100644 array.rb
create mode 100644 each.rb
create mode 100644 ejer3.rb
create mode 100644 ejer4.rb
create mode 100644 ejer5.rb
create mode 100644 ejer6.rb
create mode 100644 ejer7.rb
create mode 100644 ejercicio1real.rb
create mode 100644 ejercicio2real.rb
joysi@debian:~/trab2$

Terminal 16 de ago 16:58

joysi@debian:~/trab2$ ls
array.rb  ejer3.rb  ejer5.rb  ejer7.rb          ejercicio2real.rb
each.rb   ejer4.rb  ejer6.rb  ejercicio1real.rb
joysi@debian:~/trab2$ git status
En la rama master
nada para hacer commit, el árbol de trabajo está limpio
joysi@debian:~/trab2$
```

```
Terminal 16 de ago 17:01
joysi@debian: ~/trab2

En la rama master
nada para hacer commit, el árbol de trabajo está limpio
joysi@debian:~/trab2$ ls
array.rb  ejer3.rb  ejer5.rb  ejer7.rb  ejercicio2real.rb
each.rb   ejer4.rb  ejer6.rb  ejercicio1real.rb
joysi@debian:~/trab2$ git status
En la rama master
nada para hacer commit, el árbol de trabajo está limpio
joysi@debian:~/trab2$ git log
commit 4ab4dcff112faf981b98f96e8af230d8316fd6c1 (HEAD -> master)
Author: Joysi Soza <sozajoyssi14@gmail.com>
Date:   Fri Aug 16 16:57:14 2024 -0600

    creacion de 7 ejemplos y 2 ejercicios
joysi@debian:~/trab2$ git config --list
user.name=Joysi Soza
user.mail=sozajoyssi14@gmail.com
user.email=sozajoyssi14@gmail.com
core.editor=code
core.autocrlf=input
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
joysi@debian:~/trab2$
```

## 1. Array.

- 1.1. En el directorio ruby, crear un programa en Ruby y asignar a un array los días de la semana, para luego imprimirlos por pantalla.

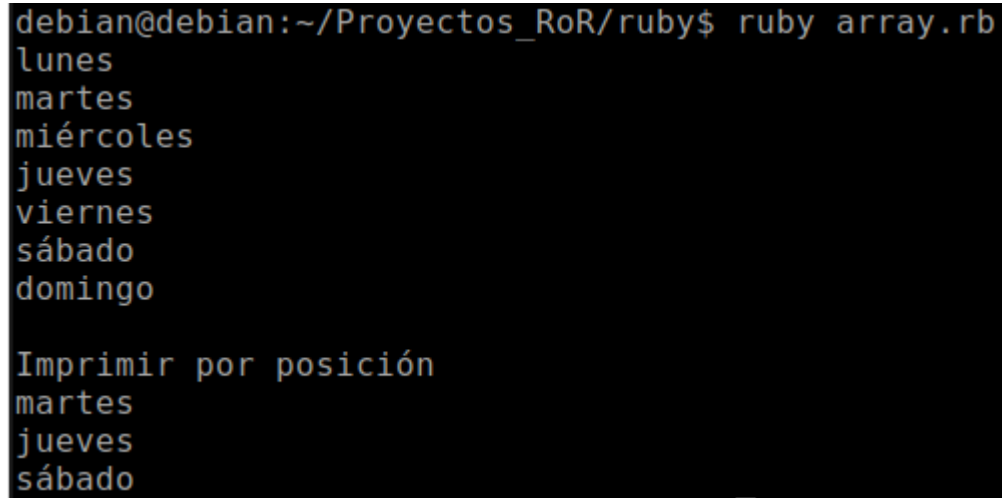
```
array.rb - trab2 - Visual Studio Code
archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda
EXPLORADOR  ...
TRAB2
  array.rb  M
  each.rb
  ejer3.rb
  ejer4.rb
  ejer5.rb
  ejer6.rb

array.rb
1  #array
2  semana=["lunes","martes","miércoles","jueves","viernes","sábada"]
3  puts semana
4  puts "\nImprimir por posición"
5  puts semana [1]
6  puts semana [3]
7  puts semana [5]
```

- 1.2. Ejecutar el programa en el terminal y analizar lo que imprime.



```
PROBLEMAS  SALIDA  TERMINAL  PUERTOS  CONSOLA DE DEPURACIÓN  bash + v [ ] [x] ...  
  
Imprimir por posición  
lunes  
martes  
jueves  
sábado  
root@debian: /home/joyasi/trab2#
```



```
debian@debian: ~/Proyectos_RoR/ruby$ ruby array.rb  
lunes  
martes  
miércoles  
jueves  
viernes  
sábado  
domingo  
  
Imprimir por posición  
martes  
jueves  
sábado
```

Como se observa en la figura 23, cada uno de los datos del array tiene una posición dentro de él, pudiéndose llamar a cada uno de ellos, con el simple hecho de escribir en qué posición se encuentra el dato que se quiere mostrar por pantalla.

```
#array
semana=["lunes","martes","miércoles","jueves","viernes","sábado","domingo"]

puts semana

puts "\nImprimir por posición"
puts semana [1]
puts semana [3]
puts semana [5]
```

## 2. Método each

2.1. 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.

```
semana=["lunes","martes","miércoles","jueves","viernes","sábado","domingo"]

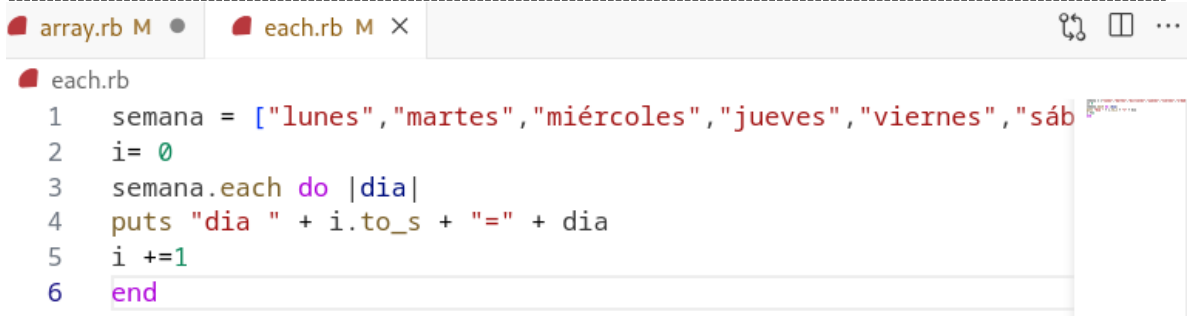
i= 0

semana.each do |dia|

    puts "dia " + i.to_s + "=" + dia

    i +=1

end
```



```
each.rb
1  semana = ["lunes", "martes", "miércoles", "jueves", "viernes", "sáb
2  i= 0
3  semana.each do |dia|
4  puts "dia " + i.to_s + "=" + dia
5  i +=1
6  end
```

La variable i sólo se utiliza como un contador para mostrar que dato es el almacenado en cada posición del array.

## 2.2. Ejecutar el programa y verificar su funcionamiento

```
debian@debian:~/Proyectos_RoR/ruby$ ruby metodo_each.rb
dia 0=lunes
dia 1=martes
dia 2=miércoles
dia 3=jueves
dia 4=viernes
dia 5=sábado
dia 6=domingo
```

El método `each` es el método más utilizado al momento de recorrer un array en Ruby, itera cada posición que existe en el array de manera ordenada.



```
PROBLEMAS  SALIDA  TERMINAL  PUERTOS  ...  
bash + - [ ] [x] ... ^ X  
  
root@debian:/home/joysi/trab2# ruby each.rb  
dia 0=lunes  
dia 1=martes  
dia 2=miércoles  
dia 3=jueves  
dia 4=viernes  
dia 5=sábado  
dia 6=domingo
```

### 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.

3.1. A continuación, se deberá realizar un programa en el que se utilicen algunos de los métodos antes mencionados.

```
semana = ["lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo"]

puts "Array en Ruby"
puts semana

puts "\nMétodo to_s"
puts semana.to_s

puts "\nMétodo join"
puts semana.join(",")

puts "\nMétodo first"
puts semana.first

puts "\nMétodo last"
puts semana.last

puts "\nMétodo length"
puts semana.length
```



```
ejer3.rb - trab2 - Visual Studio Code

Ir Ejecutar Terminal Ayuda

array.rb M  each.rb M  ejer3.rb M X

ejer3.rb
1  semana = ["lunes", "martes", "miércoles", "jueves", "viernes", "sáb
2  puts "Array en Ruby"
3  puts semana
4  puts "\nMétodo to_s"
5  puts semana.to_s
6  puts "\nMétodo join"
7  puts semana.join(",")
8  puts "\nMétodo first"
9  puts semana.first
10 puts "\nMétodo last"
11 puts semana.last
12 puts "\nMétodo length"
13 puts semana.length
```

- 3.2. Ejecutar el programa y verificar el funcionamiento, es importante ver cómo se comporta cada uno de los métodos con respecto al array.

```
debian@debian:~/Proyectos_RoR/ruby$ ruby metodo_array.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
```

```
PROBLEMAS  SALIDA  TERMINAL  PUERTOS  ...
bash + v [ ] [ ] ... ^ x

root@debian:/home/joysi/trab2# ruby ejer3.rb
Array en Ruby
lunes
martes
miércoles
jueves
viernes
sábado
domingo

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

```
PROBLEMAS  SALIDA  TERMINAL  PUERTOS  ...  
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 los arrays.

```
semana = ["lunes","martes","miércoles","jueves","viernes","sábado","domingo"]  
  
puts "Array completo"  
puts semana.to_s  
  
puts "\nMétodo POP"  
puts semana.pop  
  
puts "\nMétodo length"  
puts semana.length  
  
puts "\nÚltimo dato"  
puts semana.last  
  
puts "\nMétodo PUSH"  
puts semana.push "final"  
  
puts "\nTamaño nuevo"  
puts semana.length
```



PORTAMIENTOS ENTRE AMBOS. EN RELACION A SU USO SOBRE LOS ARRAYS.

ejer4.rb - trab2 - Visual Studio Code

Ir Ejecutar Terminal Ayuda

array.rb M each.rb M ejer3.rb M ejer4.rb X

ejer4.rb

```
1 semana = ["lunes", "martes", "miércoles", "jueves", "viernes", "sáb
2 puts "Array completo"
3 puts semana.to_s
4 puts "\nMétodo POP"
5 puts semana.pop
6 puts "\nMétodo length"
7 puts semana.length
8 puts "\nÚltimo dato"
9 puts semana.last
10 puts "\nMétodo PUSH"
11 puts semana.push "final"
12 puts "\nTamaño nuevo"
13 puts semana.length
```

PROBLEMAS SALIDA TERMINAL PUERTOS ...

bash + ▾

Al ejecutar el programa se observa que el método **pop** extrae el último dato dentro del array y lo elimina del conjunto, a diferencia del método **push**, que lo que hace es insertar el valor del dato indicado al final del conjunto.

```
debian@debian:~/Proyectos_RoR/ruby$ ruby push_pop.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
```

```
Visual Studio Code
ejer4.rb - trab2 - Visual Studio Code

Ejecutar Terminal Ayuda
PROBLEMAS SALIDA TERMINAL PUERTOS ...
bash + - [ ] [ ] ... - x

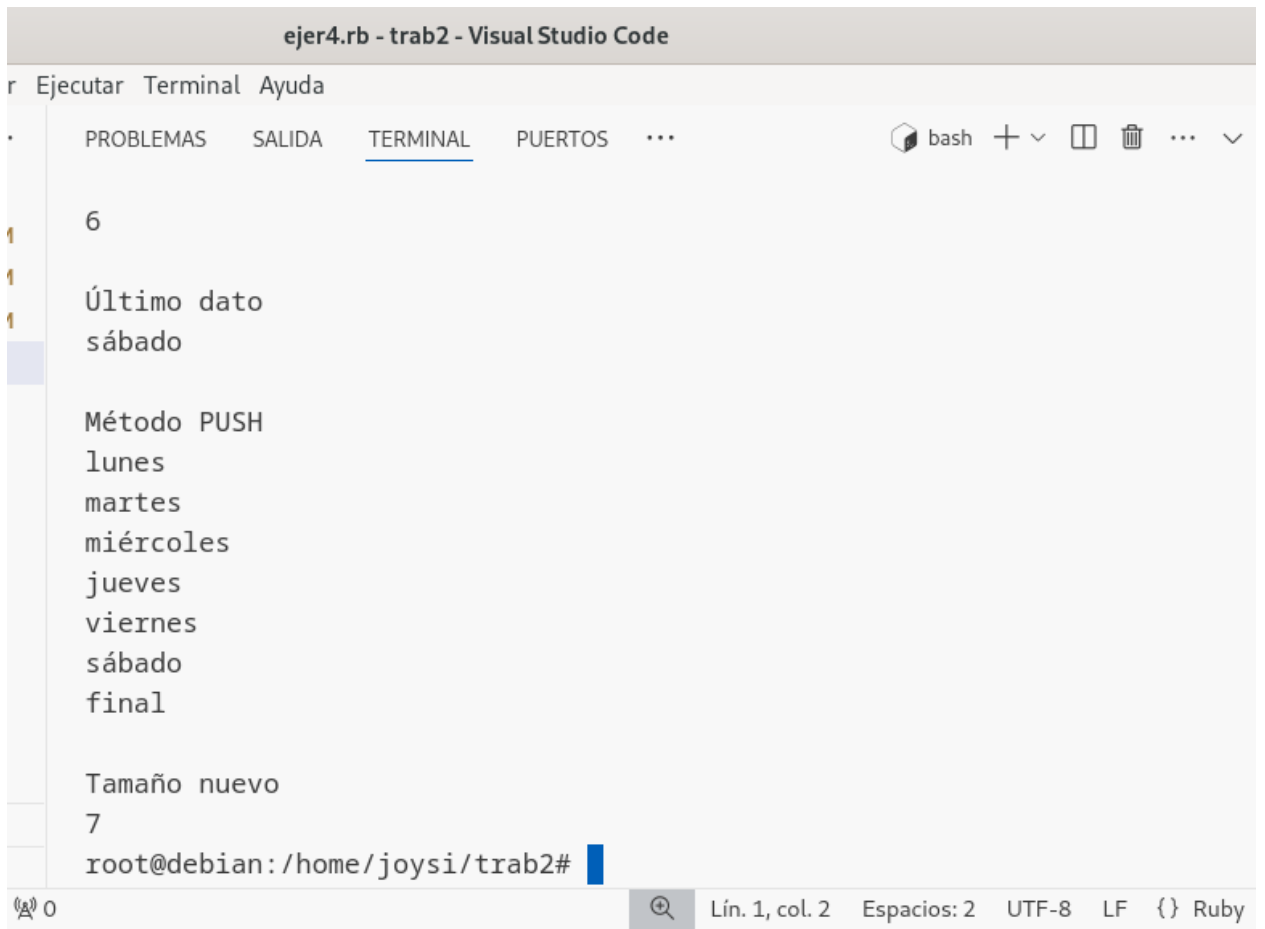
root@debian:/home/joysi/trab2# ruby ejer4.rb
Array completo
["lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo"]

Método POP
domingo

Método length
6

Último dato
sábado

Método PUSH
lunes
martes
miércoles
```



```
ejer4.rb - trab2 - Visual Studio Code
r Ejecutar Terminal Ayuda
PROBLEMAS SALIDA TERMINAL PUERTOS ...
bash + - [ ] [x] ...
6
Último dato
sábado

Método PUSH
lunes
martes
miércoles
jueves
viernes
sábado
final

Tamaño nuevo
7
root@debian:/home/joysi/trab2#
```

0 Lín. 1, col. 2 Espacios: 2 UTF-8 LF {} Ruby

#### 4. Métodos propios

4.1 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.

```
#definición de métodos
#método con parámetros
def edad (año_naciento, año_actual)

    edad = año_actual.to_i - año_naciento.to_i
    puts "\nTu edad actual es #{edad} años"
end
#método sin parámetros
def nombre
    puts "Ingrese su nombre"
    nombre = gets.chomp

    if nombre.downcase
        nombre = nombre.upcase
    else
        nombre = nombre
    end
    puts "Bienvenido #{nombre}"
end

#Invocación de los métodos
nombre
puts "Ingrese su año de nacimiento"
año_naciento = gets.chomp

puts "Ingrese el año actual"
año_actual = gets.chomp

edad(año_naciento, año_actual)
```

ay.rb M • each.rb M ejer3.rb M ejer4.rb M ejer5.i

ejer5.rb

```
3 def edad (año_nacimiento, año_actual)
4   def nombre
5   end
6 end
7
8
9 puts "Ingrese su nombre"
10 nombre = gets.chomp
11 if nombre.downcase
12 nombre = nombre.upcase
13 else
14 nombre = nombre
15 end
16 puts "Bienvenido #{nombre}"
17 end
18 #Invocación de los métodos
19 nombre
20 puts "Ingrese su año de nacimiento"
21 año_nacimiento = gets.chomp
22 puts "Ingrese el año actual"
23 año_actual = gets.chomp
24 edad(año_nacimiento, año_actual)
```

Como se puede observar se han definido dos métodos, uno llamado **nombre** que no recibe parámetros y el otro llamado **edad**, que recibe dos parámetros y que será el encargado de calcular la edad de una persona.

```
debian@debian:~/Proyectos_RoR/ruby$ ruby metodos_propios.rb
Ingrese su nombre
María
Bienvenido MARÍA
Ingrese su año de nacimiento
1995
Ingrese el año actual
2018
Tu edad actual es 23 años
```

```
root@debian:/home/joysi/trab2# ruby ejer5.rb
```

```
Ingrese su nombre
```

```
Joysi
```

```
Bienvenido JOYSI
```

```
Ingrese su año de nacimiento
```

```
2004
```

```
Ingrese el año actual
```

```
2024
```

```
Tu edad actual es 20 años
```

## 5. Hash

5.1. 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.

```
colorHash = {}

colorHash['rojo '] = '#FF0000'
colorHash['verde'] = '#008000'
colorHash['azul '] = '#0000FF'

colorHash.each do |tipoCodigo, color|
  puts tipoCodigo + ' : ' + color
end
```



ejer6.rb - trab2 - Visual Studio Code

er Ir Ejecutar Terminal Ayuda

... ejer4.rb M ejer5.rb M ejer6.rb M X ejer7.rb M ejer8.r

ejer6.rb

```
1 colorHash = {}
2 colorHash['rojo '] = '#FF0000'
3 colorHash['verde'] = '#008000'
4 colorHash['azul '] = '#0000FF'
5 colorHash.each do |tipoCodigo, color|
6   puts tipoCodigo + ' : ' + color
7 end
```

5.2. Ejecutar el programa en el terminal para obtener la salida.

```
debian@debian:~/Proyectos_RoR/ruby$ ruby hash.rb
rojo : #FF0000
verde : #008000
azul : #0000FF
```

```
root@debian:/home/joysi/trab2# ruby ejer6.rb
rojo : #FF0000
verde : #008000
azul : #0000FF
```

En los hashes, cada dato almacenado se guarda con un nombre que se conoce como una clave, esta clave puede ser textos o número y es por medio del cual se puede identificar cada uno de los datos que pertenecen al hash; como se puede observar en la **figura 28**, se imprime cada dato del hash por medio de su clave.

5.3. Para ver de otra manera el funcionamiento, crear un nuevo programa hash\_2.rb y agregar el código.

```
user = {}  
user = { :name => "Juan Pérez", :email => "JuanP@example.com" }  
  
puts "Nombre de usuario: #{user[:name]} "  
puts "\nCorreo: #{user[:email]}"
```



Al ejecutar el programa se observa cómo se hace referencia a los datos del usuario, haciendo uso de la clave para poder mostrarlos por pantalla.

```
debian@debian:~/Proyectos_RoR/ruby$ ruby hash_2.rb  
Nombre de usuario: Juan Pérez  
  
Correo: JuanP@example.com
```

```
root@debian:/home/joysi/trab2# ruby ejer7.rb  
Nombre de usuario: Juan Pérez  
  
Correo: JuanP@example.com
```



## 6. Clases

- 6.1. 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.

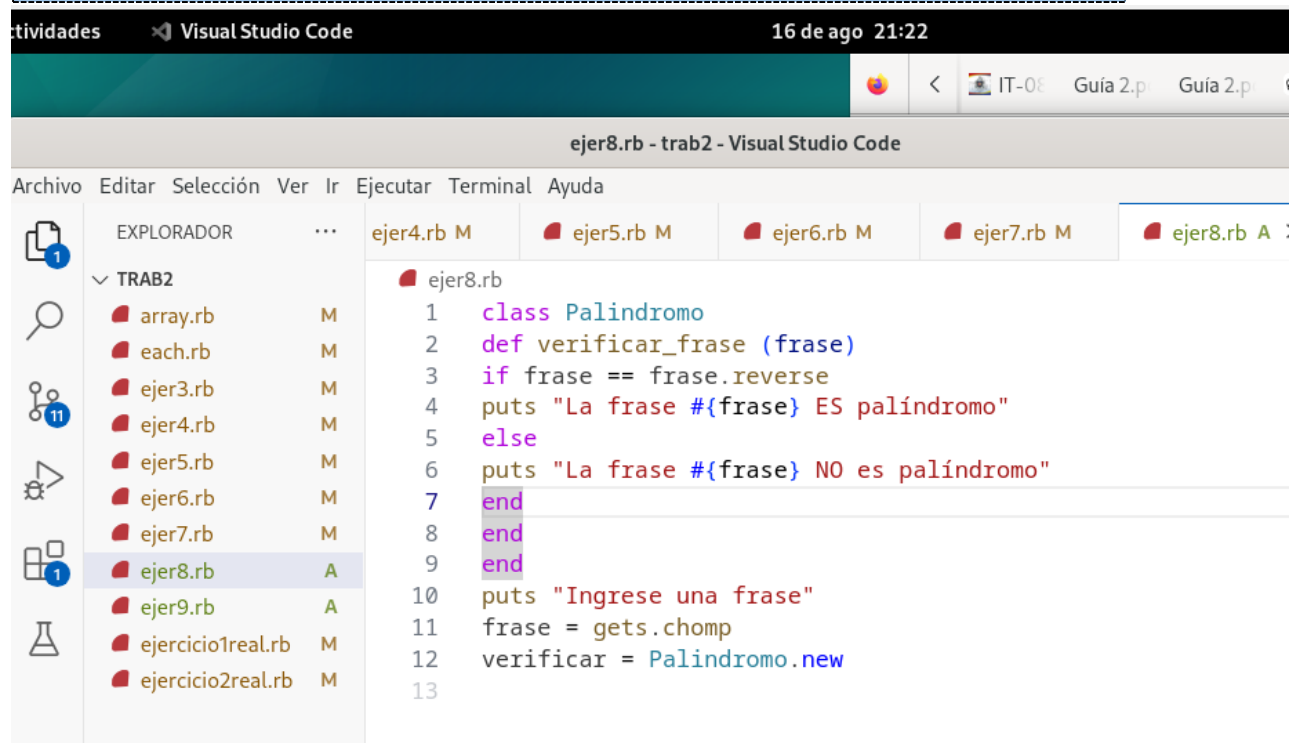
```
class Palindromo

  def verificar_frase (frase)

    if frase == frase.reverse
      puts "La frase #{frase} ES palíndromo"
    else
      puts "La frase #{frase} NO es palíndromo"
    end
  end
end

puts "Ingrese una frase"
frase = gets.chomp

verificar = Palindromo.new
```



- 6.2. Ejecutar el programa e ingresar la palabra "level" para verificar su correcto funcionamiento.

```
debian@debian:~/Proyectos_RoR/ruby$ ruby clases.rb
Ingrese una frase
level
La frase level ES palíndromo
```

```
Correo: JuanP@example.com
root@debian:/home/joysi/trab2# ruby ejer8.rb
Ingrese una frase
kevel
root@debian:/home/joysi/trab2#
```

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.

7.1. 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:

```
class Dado

  def initialize
    rodar
  end

  def rodar
    @numero_mostrar = 1 + rand(6)
  end

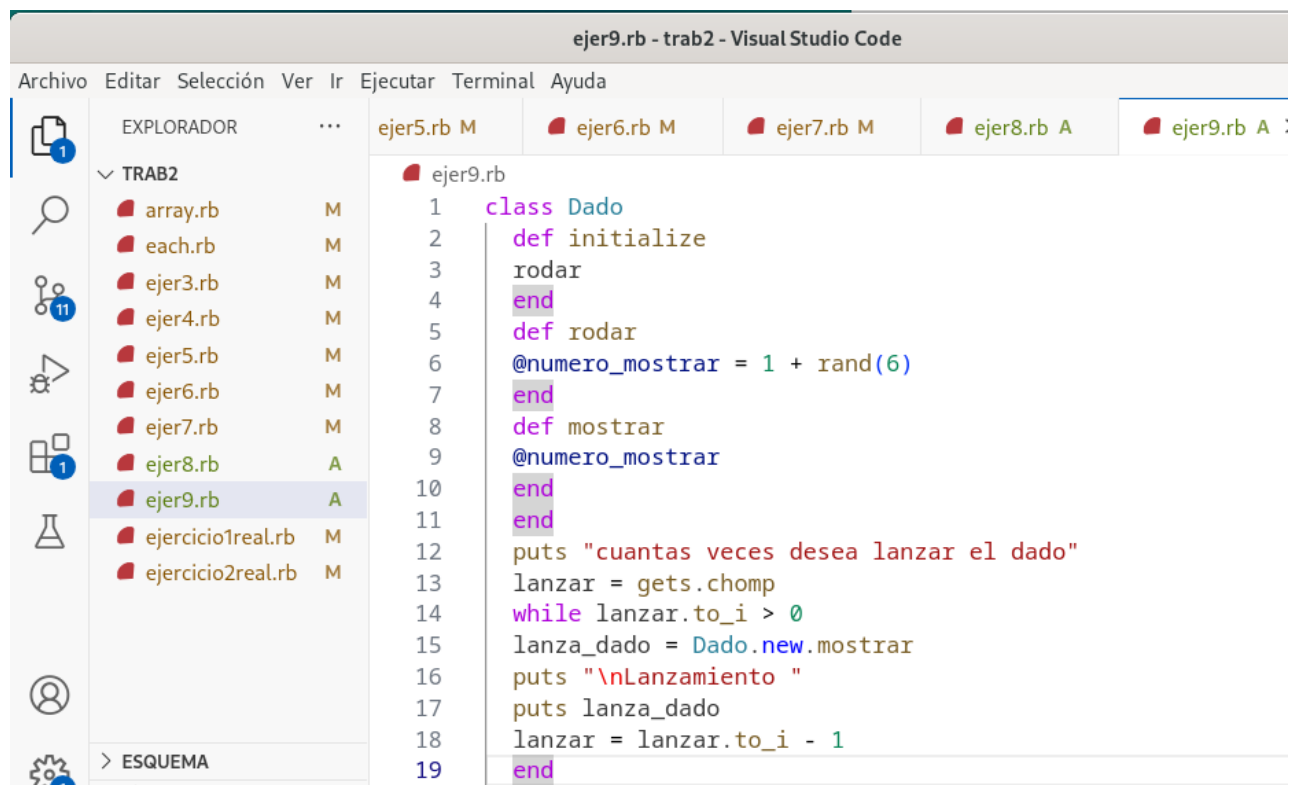
  def mostrar
    @numero_mostrar
  end

  puts "cuantas veces desea lanzar el dado"
  lanzar = gets.chomp

  while lanzar.to_i > 0

    lanza_dado = Dado.new.mostrar
    puts "\nLanzamiento "
    puts lanza_dado
    lanzar = lanzar.to_i - 1

  end
end
```



The screenshot shows the Visual Studio Code interface with the file 'ejer9.rb' open. The Explorer sidebar on the left shows a project named 'TRAB2' containing several files: 'array.rb', 'each.rb', 'ejer3.rb', 'ejer4.rb', 'ejer5.rb', 'ejer6.rb', 'ejer7.rb', 'ejer8.rb', 'ejer9.rb', 'ejercicio1real.rb', and 'ejercicio2real.rb'. The file 'ejer9.rb' is selected and highlighted. The main editor area displays the Ruby code for 'ejer9.rb', which is identical to the code block provided above. The code defines a 'Dado' class with methods for initialization, rolling, and displaying the result, along with a loop to ask the user how many times to roll the die.

```
ejer9.rb - trab2 - Visual Studio Code

Archivo  Editor  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda

EXPLORADOR  ...  ejer5.rb M  ejer6.rb M  ejer7.rb M  ejer8.rb A  ejer9.rb A :

TRAB2
  array.rb M
  each.rb M
  ejer3.rb M
  ejer4.rb M
  ejer5.rb M
  ejer6.rb M
  ejer7.rb M
  ejer8.rb A
  ejer9.rb A
  ejercicio1real.rb M
  ejercicio2real.rb M

> ESQUEMA

ejercicio9.rb
1  class Dado
2    def initialize
3      rodar
4    end
5    def rodar
6      @numero_mostrar = 1 + rand(6)
7    end
8    def mostrar
9      @numero_mostrar
10   end
11  end
12  puts "cuantas veces desea lanzar el dado"
13  lanzar = gets.chomp
14  while lanzar.to_i > 0
15    lanza_dado = Dado.new.mostrar
16    puts "\nLanzamiento "
17    puts lanza_dado
18    lanzar = lanzar.to_i - 1
19  end
```

Como se observa en el código la variable **numero\_mostrar**, se utiliza en los métodos rodar y mostrar, y siempre mantiene el mismo el valor.

- 7.2. Ejecutar el programa y verificar el funcionamiento de la variable `numero_mostrar`, la cual mantiene su valor en todos los métodos hasta ser mostrada por pantalla.

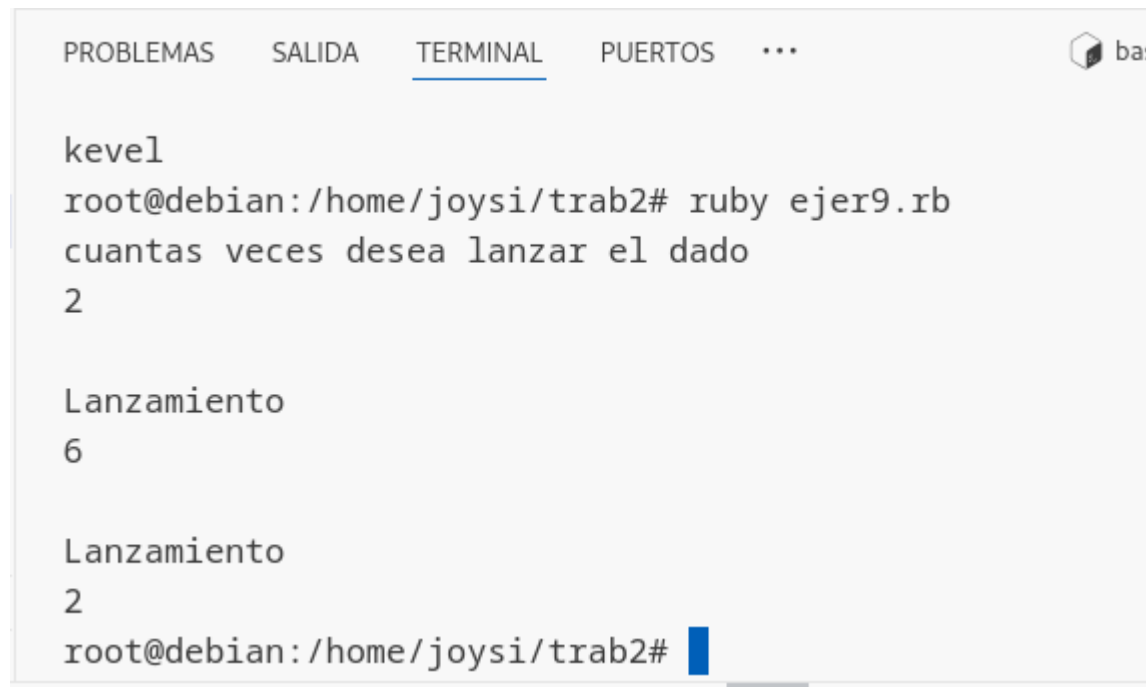
```
debian@debian:~/Proyectos_RoR/ruby$ ruby variables.rb
cuantas veces desea lanzar el dado
4

Lanzamiento
3

Lanzamiento
3

Lanzamiento
2

Lanzamiento
1
```



```
PROBLEMAS  SALIDA  TERMINAL  PUERTOS  ...  bas

kevel
root@debian:/home/joysi/trab2# ruby ejer9.rb
cuantas veces desea lanzar el dado
2

Lanzamiento
6

Lanzamiento
2
root@debian:/home/joysi/trab2#
```

### Ejercicios propuestos para ser entregados al docente

1. Realice cada uno de los enunciados de la guía, probar el funcionamiento y analizar cada uno de los programas planteados.

Podemos apreciar que este inciso se ha realizado en la parte superior.

2. 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.

```
debian@debian:~/Proyectos_RoR/ruby$ ruby asignacion2.rb
Nombre          Celular
María           2248-6559
Pedro           9845-6532
Juan            8265-4536
Alberto         7896-4514
-----
Ingrese un nombre
juan
-----
El número de celular de Juan : 8265-4536
```

■ ejercicio1real.rb

```
1  # Crear el hash con nombres y números de celular
2  contactos = {
3    'Maria' => '2248-6559',
4    'Pedro' => '9845-6532',
5    'Juan' => '8265-4536',
6    'Alberto' => '7896-4514'
7  }
8
9  loop do
10   # Mostrar el hash completo
11   puts "Hash completo:"
12   contactos.each do |nombre, celular|
13     puts "#{nombre}: #{celular}"
14   end
15
16   puts "-----"
17
18   # Solicitar al usuario que ingrese un nombre
19   print "Ingrese el nombre para buscar el celular: "
20   nombre_usuario = gets.chomp
21
22   # Convertir el nombre ingresado a título (inicial mayúscula)
23   nombre_usuario_capitalizado = nombre_usuario.capitalize
24   puts "-----"
25
26   # Buscar y mostrar el número de celular correspondiente
27   if contactos.key?(nombre_usuario_capitalizado)
28     puts "El celular de #{nombre_usuario_capitalizado} es #{co
29   else
30     puts "El nombre '#{nombre_usuario}' no se encuentra en la
31   end
32
33   # Preguntar al usuario si desea continuar o salir
34   print "¿Desea buscar otro nombre? (Presione 'S' para salir):
35   respuesta = gets.chomp.upcase
36
```

```

32
33   # Preguntar al usuario si desea continuar o salir
34   print "¿Desea buscar otro nombre? (Presione 'S' para salir):"
35   respuesta = gets.chomp.upcase
36
37   # Salir del bucle si el usuario presiona 'S'
38   break if respuesta == 'S'
39 end
40
41 puts "Gracias por usar el programa."
42

```

```

root@debian:/home/joysi/trab2# ruby ejercicio1real.rb
Hash completo:
Maria: 2248-6559
Pedro: 9845-6532
Juan: 8265-4536
Alberto: 7896-4514
-----
Ingrese el nombre para buscar el celular: Maria
-----
El celular de Maria es 2248-6559.
¿Desea buscar otro nombre? (Presione 'S' para salir): S
Gracias por usar el programa.
root@debian:/home/joysi/trab2#

```

3. 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 numero impares, se deberá mostrar a como se muestra.

```

debian@debian:~/Proyectos_RoR/ruby$ ruby suma_pares_impares.rb
Ingrese el número maximo a sumar:
15
La suma de los números PARES de 15 es:
56
La suma de los números IMPARES de 15 es:
64

```

■ ejercicio2real.rb

```
1  class Calcular
2    def initialize(numero)
3      @numero = numero.to_s
4    end
5
6    def suma_pares
7      pares = @numero.chars.select { |char| char.to_i.even? }.map(&:to_i)
8      pares.sum
9    end
10
11    def suma_impares
12      impares = @numero.chars.select { |char| char.to_i.odd? }.map(&:to_i)
13      impares.sum
14    end
15  end
16
17  # Solicitar al usuario un número
18  print "Ingrese un número: "
19  numero_usuario = gets.chomp
20
21  # Solicitar al usuario un número
22  print "Ingrese un número: "
23  numero_usuario = gets.chomp
24
25  # Crear una instancia de la clase Calcular
26  calculadora = Calcular.new(numero_usuario)
27
28  # Mostrar los resultados
29  puts "Suma de números pares: #{calculadora.suma_pares}"
30  puts "Suma de números impares: #{calculadora.suma_impares}"
31
```

```
root@debian:/home/joysi/trab2# ruby ejercicio2real.rb
Ingrese un número: 22
Suma de números pares: 4
Suma de números impares: 0
root@debian:/home/joysi/trab2#
```

**Extra:** hacer que el programa número 1 que se ejecute varias veces hasta presionar una tecla.