

UNIVERSIDAD AUTÓNOMA DE  
QUERÉTARO

PARADIGMAS DE PROGRAMACIÓN

GUÍA RUBY

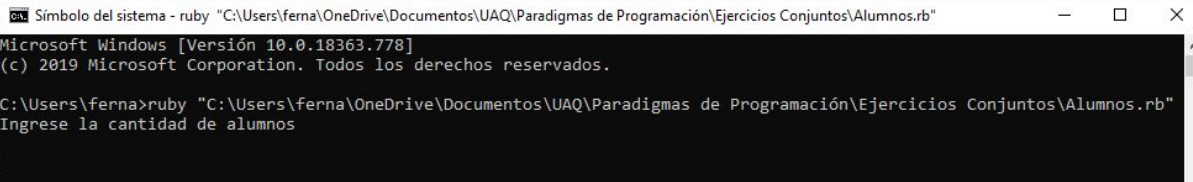
Docente: Sofía Amadís Rivera López

27/02/2020

# GUIA RUBY

Primero que nada para poder usar Ruby se necesita descargar el IDE de Ruby, se encuentra fácilmente en internet, es para poder correr el código o en dado caso crear el código, es recomendable que se desarrolle el código desde un editor externo como Sublimetext.

Lo primero que se debe de hacer es descargar Ruby, instalarlo y despues se puede usar por medio de cmd como una especie de comando para mandar a llamar un archivo de ruby, por ejemplo



```
Símbolo del sistema - ruby "C:\Users\ferna\OneDrive\Documentos\UAQ\Paradigmas de Programación\Ejercicios Conjuntos\Alumnos.rb"
Microsoft Windows [Versión 10.0.18363.778]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\ferna>ruby "C:\Users\ferna\OneDrive\Documentos\UAQ\Paradigmas de Programación\Ejercicios Conjuntos\Alumnos.rb"
Ingrese la cantidad de alumnos
```

En esta imagen se muestra que se abre el cmd y se escribe la palabra reservada Ruby, después de la palabra reservada se da un espacio y se copia la ubicación del archivo y que archivo es para así poder correrlo.

O simplemente puedes crear tu archivo ruby, dar click sobre el y abra el comand Promt de Ruby y lo corre automaticamente.

278787  
278828  
278816  
266145

Carlos Emilio Cortés Martínez  
Fernando Maya Carranza  
Diego A. Lomeli Vazquez  
Ivonne Monserrat Cruz Paz

**Rápido y sencillo:**

- Son innecesarias las declaraciones de variables
- Las variables no tienen tipo
- La sintaxis es simple y consistente
- La gestión de la memoria es automática

**Programación orientada a objetos:**

- Todo es un objeto
- Clases, herencia, métodos:

Las Clases se definen con la primera letra en Mayúsculas ya que si es minúscula la toca como “variable” y viceversa. Por ejemplo:

**class Mayusculas**

**end**

- Iteradores y cierres

**También:**

- Enteros de precisión múltiple
- Modelo de procesamiento de excepciones
- Carga dinámica
- Hilos

**Concatenación:**

La concatenación en Ruby se realiza con una “coma”(,), a diferencia de los lenguajes convencionales que utilizan el signo de “más”(+) , en este caso, al usar el signo + lo reconocerá como una función matemática, la cual es la de Suma.

:

En Ruby no es necesario el uso de algún símbolo o carácter para la finalización de sentencia, simplemente con el cambio de línea lo reconocerá como fin de sentencia, por lo cual la indentación no es importante, simplemente se recomienda para lograr un código más legible y estético.

Operadores:

- Operadores aritméticos:

\_ + - / \* \*\* %

- Operadores relacionais:

\_ == != > < >= <=

- Operadores lógicos:

\_ and or

Las variables se definen en minúsculas, por ejemplo:

a = 1

b = “Hola Mundo”

278787

278828

278816

266145

Carlos Emilio Cortés Martínez

Fernando Maya Carranza

Diego A. Lomeli Vazquez

Ivonne Monserrat Cruz Paz

Como se mencionó no se declara el tipo de variable, al momento de ingresar un dato sabe de qué tipo será, pero en el caso de cadenas de texto debe de tener “comillas dobles” ya que con comillas simples tiene distinto efecto, por ejemplo:

```
ruby> print "a\nb\nc","\n"
a
b
c
nil
ruby> print 'a\nb\nc','\n'
a\nb\nc
nil
```

Ya sea con comillas simples o dobles debe de tener comillas una cadena de texto ya que así puede distinguir más simplemente el tipo de dato de la variable.

La concatenación de las variables es con un +, pero la concatenación de un texto de puts o print y una variable se concatena con una “,”.

En el siguiente ejemplo podemos ver lo más básico para poder usar Ruby:

Selectivo

class MayorMenor **Se declara la clase**

def numeros **def, es la palabra reservada para una función def = definir (definir qué se hará en la “función”)**

puts "Ingrese un numero: " **“puts” es lo mismo que “print”**

a = gets.to\_i

**gets.to\_i lo que hace en esta parte es decirle a la variable que va a obtener los datos de el puts**

puts "Ingrese un segundo numero: "

b = gets.to\_i

c = (a+b)

**aquí vemos como primer operador aritmético “+” en esta parte el + concatena para hacer la operación dentro de la variable aunque no es necesario poner paréntesis a menos que haya más operadores dentro de la “operación”**

print "La suma de los numeros es: ", c, "\n" **Se imprime la operación con un salto de línea para el siguiente texto del if**

if(a>b)

278787

Carlos Emilio Cortés Martínez

278828

Fernando Maya Carranza

278816

Diego A. Lomeli Vazquez

266145

Ivonne Monserrat Cruz Paz

```

        print a," es mayor que ",b
    else
        print b, " es mayor que ", a
    end

```

**Poner end es importante ya que si no se declara no tendrá fin la condición y no funcionara el código**

## **CUANDO TERMINA LA FUNCIÓN SE CREA EL OBJETO PARA QUE EL CÓDIGO SE CORRA EN EL INTÉRPRETE**

```

num = MayorMenor.new()
num.numeros
gets()

```

### **Ejemplo Secuencial Ciclo WHILE**

```

class Ejercicios
  def initialize()
  end
  def ex()
    print("Ingrese el numero final ")
    n = gets.to_i
    x = 1
    while x<=n un While igual que en cualquier lenguaje a excepción de llaves o dos puntos
      print "[",x,"]"
      x = x+1;
    end
  end
end
objeto = Ejercicios.new()
objeto.ex
gets()

```

## **EJERCICIOS 4,5,9,17**

### **4.-**

```

#Declaración de variables
Descuento = 0.20
MinElegible = 1000

```

278787	Carlos Emilio Cortés Martínez
278828	Fernando Maya Carranza
278816	Diego A. Lomeli Vazquez
266145	Ivonne Monserrat Cruz Paz

```

puts("Dame la cantidad que compro el cliente: ")

#Transforma input a una integral
compra = gets.chomp.to_i

# Obtiene el residuo del total de la compra con el mínimo para que aplique el descuento.
Contador = compra / MinElegible

#Si el residuo es mayor a 1... entonces
if Contador > 1

#Se obtiene el descuento total
totalDescuento = Descuento * Contador

#Se saca el valor a restar
cantidadARestar = compra * totalDescuento


#Se realiza el cálculo del total
total = compra - cantidadARestar
puts ("Tu total con descuento es de: ")
puts (total)

#Si el residuo es menor a 1.. entonces se imprime el total de compra sin el descuento.
elsif Contador <= 1
  puts ("No entra la promoción, su total es de: ")
  puts (compra)

end

```

## 9.-

#Declaración de variables

```

$costo1 = 5
$costo2 = 4
$costo3 = 3
$costo4 = 2

```

278787	Carlos Emilio Cortés Martínez
278828	Fernando Maya Carranza
278816	Diego A. Lomeli Vazquez
266145	Ivonne Monserrat Cruz Paz

```

$costo = 0
$horas = 0
class Estacionamiento
  def tarifas()
    puts "Ingrese las horas en el Estacionamiento: "
    $horas = gets.to_i #Recibe el input y lo convierte a int
    case es como un switch en algunos otro lenguajes, pero en vez de
    switch es case y en vez de case es when
    case $horas
    when 1..2
      $costo = ($horas*$costo1)
      print "Es costo es de: ", $costo
    when 3..5
      $costo1 = 10
      if $horas ==3
        $costo2 = (4+$costo1)
        puts "El costo es de: ", $costo2
      end
      if $horas ==4
        $costo2 = (8+$costo1)
        puts "El costo es de: ", $costo2
      end
      if $horas ==5
        $costo2 = (12+$costo1)
        puts "El costo es de: ", $costo2
      end
    end
    when 6..10
      $costo2 = 22
      if $horas==6
        $costo3 = ($costo2 + 3)
        puts "El costo es de: ", $costo3
      end
      if $horas==7
        $costo3 = ($costo2 + 6)
        puts "El costo es de: ", $costo3
      end
      if $horas==8
        $costo3 = ($costo2 + 9)
        puts "El costo es de: ", $costo3
      end
      if $horas==9
        $costo3 = ($costo2 + 12)
        puts "El costo es de: ", $costo3
      end
    end
  end
end

```

278787	Carlos Emilio Cortés Martínez
278828	Fernando Maya Carranza
278816	Diego A. Lomeli Vazquez
266145	Ivonne Monserrat Cruz Paz

```

        if $horas==10
            $costo3 = ($costo2 + 15)
            puts "El costo es de: ", $costo3
        end
    when 10..
        if $horas>10
            $costo4 = ($horas*2)
            puts "El costo es de: ", $costo4
        end
    else
        print"Error en la variable"
    end
end
end
end
auto = Estacionamiento.new()
auto.tarifas
gets()

```

```

class Empleado
    def horasT
        puts "Cuántas horas trabajadas a la semana: "
        htrabajadas = gets.to_i
        if htrabajadas<=40
            salario = (htrabajadas*16)
            puts "Tu salario semanal es de: ", salario
        end
        if htrabajadas>=41
            extras = ((htrabajadas - 40)*20) Uso de operadores algorítmicos
            salario = ((40*16) + extras)
            puts "Tu salario semanal es de: ", salario
        end
    end
end
end

```

### **creación del objeto**

```

horasS = Empleado.new()
horasS.horasT
gets()

```

### **17.-**

```

class Inter
    def medio
        print"Ingresa el Primer numero \n"
        a=gets.to_i
    end
end

```

278787	Carlos Emilio Cortés Martínez
278828	Fernando Maya Carranza
278816	Diego A. Lomeli Vazquez
266145	Ivonne Monserrat Cruz Paz



```

        print "Ingresa el Segundo numero \n"
        b = gets.to_i
        print "Ingresa el Tercer numero \n"
        c = gets.to_i
        if (a > b && b > c) || (c > b && b > a)
            print b, " Es el numero intermedio"
        end
        if (b > a && a > c) || (c > a && a > b)
            print a, " Es el numero intermedio"
        end
        if (a > c && c > b) || (b > c && c > a)
            print c, " Es el numero intermedio"
        end
    end
end
nums = Inter.new()
nums.medio
gets()

```

## Arrays

- Un array puede ser creado con el constructor `[]` y separándolos por coma.
- Puede contener diferentes tipos de datos. (Objetos, Integer, string, etc).

- Puede ser creado con `.new`, o con dos argumentos; El tamaño, y el dato default

```

ary = Array.new      #=> []
Array.new(3)         #=> [nil, nil, nil]
Array.new(3, true)   #=> [true, true, true]

```

- Elementos en el array pueden ser obtenidos con el método `"#[]"`. Puede usarse con un argumento entero, (índice numerico), un par de argumentos (principio y fin).

278787  
278828  
278816  
266145

Carlos Emilio Cortés Martínez  
Fernando Maya Carranza  
Diego A. Lomeli Vazquez  
Ivonne Monserrat Cruz Paz

```

arr = [1, 2, 3, 4, 5, 6]
arr[2]    #=> 3
arr[100]  #=> nil
arr[-3]   #=> 4
arr[2, 3] #=> [3, 4, 5]
arr[1..4] #=> [2, 3, 4, 5]
arr[1..-3] #=> [2, 3, 4]

```

Métodos más importantes de los Arrays.

Método	Explicación
Arr.at(int)	Recupera elemento en el índice señalado
arr.fetch(int)	Para señalar un error por índices fuera de los límites del array, o para obtener un valor default.
arr.first	Regresa el primer elemento del array
arr.last	Regresa el último elemento del array
arr.take(int)	Regresa los elementos hasta el índice indicado.
arr.drop(int)	Regresa los elementos después del índice indicad.
arr.length/.count/.size	Regresa el tamaño del array.
arr.empty?	Regresa true or false si el elemento contiene elementos.
arr.include?('xxxx')	Regresa si un ítem está incluido en el Array.

### Agregar elementos al array:

278787	Carlos Emilio Cortés Martínez
278828	Fernando Maya Carranza
278816	Diego A. Lomeli Vazquez
266145	Ivonne Monserrat Cruz Paz

Se utiliza push ó <<.

- arr.push(5).
- arr<<6.

Con arr.insert(posición, elemento) se puede indicar el índice donde se desea agregar el elemento.

Con arr.unshift(elemento) se agrega el elemento al principio del array.

#### Quitar elementos del array:

- El método arr.pop elimina el último elemento del array y lo regresa.
- Para regresar y eliminar el primer elemento del array se utiliza arr.shift
- Para eliminar un elemento en un índice específico se utiliza arr.delete\_at(índice)
- Para eliminar un elemento en particular en el array se utiliza arr.delete(elemento)

## RECURSIVIDAD

Recurrencia, recursión o recursividad es la forma en la cual se especifica un proceso basado en su propia definición. La recursión tiene esta característica discernible en términos de autorreferencialidad, autopoiesis, fractalidad, o, en otras palabras, construcción a partir de un mismo tipo.

Básicamente lo que la recursividad es; es que si creas una funcion, en la misma función se tiene que mandar a llamar a sí misma.

Para dar a entender mejor la recursividad tenemos algunos ejemplos para mostrarles.

### La secuencia Fibonacci

```
def fibonacci( n ) AQUÍ CREAMOS LA FUNCIÓN  
  return n if n <= 1 HACEMOS RECURSIVIDAD  
  fibonacci( n - 1 ) + fibonacci( n - 2 ) LA IMPLEMENTAMOS CON LA FÓRMULA  
FIBONACCI
```

end

**ASÍ ES COMO SE MOSTRARÁ LA SECUENCIA DEPENDIENDO DE EL  
NUMERO A LLEGAR**

#fibonacci(5) = fibonacci(4) + fibonacci(3)

278787  
278828  
278816  
266145

Carlos Emilio Cortés Martínez  
Fernando Maya Carranza  
Diego A. Lomeli Vazquez  
Ivonne Monserrat Cruz Paz

```
#fibonacci(5) = fibonacci(3) + fibonacci(2) + fibonacci(2) + fibonacci(1)
#fibonacci(5) = fibonacci(2) + fibonacci(1) + fibonacci(1) + fibonacci(0) + fibonacci(1)
+ fibonacci(0) + fibonacci(1)
#fibonacci(5) = fibonacci(1) + fibonacci(0) + fibonacci(1) + fibonacci(1) + fibonacci(0)
+ fibonacci(1) + fibonacci(0) + fibonacci(1)
```

```
#fibonacci(5) = 5;
```

```
puts "Dame el numero máximo de la secuencia a llegar "
n = gets.chomp.to_i
for i in 0..n
  puts fibonacci(i)
end

gets()
```

### Triángulo de Pascal

```
def factorial(num) // Función que calcula el factorial con recursividad
  if num == 0
    return 1
  else
    num *= factorial(num - 1)
  end
end
```

```
def find_num(n, k)
  result = factorial(n) / (factorial(k) * factorial(n - k)) //Funcion que usa y ejecuta la
formula para la creacion del triangulo
end
```

```
def pascale(num) //Funcion que crea el triangulo
  i = 0
  scale = 75 //Variable que sirve para dar la figura de triangulo
  while i <= num
    new_arr = []
```

278787	Carlos Emilio Cortés Martínez
278828	Fernando Maya Carranza
278816	Diego A. Lomeli Vazquez
266145	Ivonne Monserrat Cruz Paz

```
(0..i).map {|x| new_arr << find_num(i, x)} // Se crea y se guarda un nuevo array con los numeros del triangulo
```

```
p new_arr.to_s.rjust(50 + scale) //Se imprime el array nuevo en forma de triangulo
```

```
i += 1
```

```
scale += 1
```

```
end
```

```
end
```

```
puts "Dame la fila a la que quiere llegar: " //Input para el numero de filas que se desee
```

```
n = gets.chomp.to_i;
```

```
pascale(n)
```

```
gets()
```

### **Decimal a Hexadecimal**

**def hexa(n,s) Función recursiva donde se convertirán los decimales a hexadecimales**

```
return s if n == 0
```

```
residuo = n % 16
```

```
division = n/16
```

```
if residuo > 9
```

```
case residuo
```

```
when 10
```

```
s+="A"
```

```
when 11
```

```
s+="B"
```

```
when 12
```

```
s+="C"
```

```
when 13
```

```
s+="D"
```

```
when 14
```

```
s+="E"
```

```
when 15
```

```
s+="F"
```

278787

278828

278816

266145

Carlos Emilio Cortés Martínez

Fernando Maya Carranza

Diego A. Lomeli Vazquez

Ivonne Monserrat Cruz Paz

```

        end
    else
        s+=residuo.to_s
    end
    hexa(division,s)
end
Se imprimen los datos
puts"Ingresa un numero"
a = gets.chomp.to_i
puts "la conversion a hexadecimal es "
puts hexa(a,"").reverse

```

## Algoritmos de Ordenamiento en Ruby

En el caso de estos algoritmos veremos 3 tipos de implementación:

1. Burbuja
2. Quick
3. Shell

### Burbuja

La idea detrás del ordenamiento de burbujas es que los elementos más grandes "burbujearán" hacia el final y los elementos más pequeños "burbujearán" hacia el principio hasta que todos los elementos estén en su ubicación correcta. Esto ocurre a través del intercambio repetido de elementos adyacentes si están en el orden incorrecto. Este algoritmo comienza al principio de la matriz, compara cada elemento con el elemento inmediatamente a la derecha del mismo y realiza un intercambio si los elementos están fuera de orden entre sí. Al final de cada pasada a través de la matriz, nunca es una garantía que todos los elementos estén ordenados o que todos los elementos no estén ordenados porque el algoritmo solo compara pares de elementos a la vez. ¿Cómo sabremos si los elementos de la matriz están ordenados?

Aquí creamos un intercambio de variable que es un indicador para ver si se produjo o no un intercambio durante un paso dado de la matriz. Si el intercambio devuelve falso al final del pase, eso significa que no se produjeron intercambios porque todo estaba en orden y, por lo tanto, la matriz está ordenada.

278787  
278828  
278816  
266145

Carlos Emilio Cortés Martínez  
Fernando Maya Carranza  
Diego A. Lomeli Vazquez  
Ivonne Monserrat Cruz Paz

```

1  def bubble_sort(array)
2    return array if array.size <= 1
3    swap = true
4    while swap
5      swap = false
6      (array.length - 1).times do |x|
7        if array[x] > array[x+1]
8          array[x], array[x+1] = array[x+1], array[x]
9          swap = true
10       end
11     end
12   end
13   array
14 end
15 gets()

```

1. bubble\_sort toma un solo parámetro de matriz.
2. Si el tamaño de la matriz es 1 o 0, devuelva la matriz; de manera predeterminada, se ordena una matriz vacía o una matriz con un elemento.
3. Cree la variable de "swap" y configúralo como verdadera de manera predeterminada
4. Cree un ciclo "while" que se ejecutará siempre que el "swap" sea verdadero.
5. Establezca swap = falso ya que inmediatamente después del comienzo de su ciclo, no ha habido intercambios.
6. En el bucle, recorre cada elemento de la matriz y verifique si el elemento x es mayor que el elemento al lado x + 1. Si es así, cambie el valor de x con el valor de x + 1 y establezca el valor de swap en true ya que hicimos un intercambio.
7. El bucle se repite hasta que cada elemento esté en orden y el valor de intercambio permanezca en falso. El bucle terminará y se devolverá la matriz.

## Quick

Este fue probablemente el segundo tipo más difícil de entender. El algoritmo elige un pivote, un elemento aleatorio en la matriz, y clasifica los elementos a su alrededor en función de si un elemento es mayor o menor que un pivote. Después de la primera pasada cuando cada valor menor que el pivote está en el lado izquierdo y cada valor mayor que el pivote está en el lado derecho, nos dividimos en dos subconjuntos y aplicamos una clasificación rápida a cada mitad (elija un nuevo pivote, compare elementos, se dividen en dos subconjuntos).

278787  
278828  
278816  
266145

Carlos Emilio Cortés Martínez  
Fernando Maya Carranza  
Diego A. Lomeli Vazquez  
Ivonne Monserrat Cruz Paz

```

1
2 def quick_sort(array)
3   return array if array.length <= 1
4   pivot = array.delete_at(rand(array.length))
5
6   left = Array.new
7   right = Array.new
8
9   array.each do |x|
10    if x <= pivot
11      left << x
12    else
13      right << x
14    end
15  end
16
17  return *quick_sort(left), pivot, *quick_sort(right)
18
19 end

```

Shell

def exchange e, list #Para intercambiar los valores

```

(0...(list.length-e)).each do |j|
  if list[j] >= list[j+e]
    list[j], list[j+e] = list[j+e], list[j]
  end
end
end
end

```

def shell\_sort(list) Función para definir el shell

```

d = list.length
return -1 if d == 0
(0...list.length).each do |i| #ciclo para dividir en 2
  d = d / 2
  puts "d:#{d}" #Solo es un índice para las repeticiones
  exchange(d, list)
  puts list.inspect
  if d == 1
    exchange(d, list)
    break
  end
end
end
list
end

```

arr = [4,6,2,8,1,3,9,5,7] arreglo con el que se va a trabajar

```

p shell_sort(arr)
gets()

```

278787

Carlos Emilio Cortés Martínez

278828

Fernando Maya Carranza

278816

Diego A. Lomeli Vazquez

266145

Ivonne Monserrat Cruz Paz



El algoritmo Shell es una mejora de la ordenación por inserción, donde se van comparando elementos distantes, al tiempo que se los intercambian si corresponde. A medida que se aumentan los pasos, el tamaño de los saltos disminuye; por esto mismo, es útil tanto como si los datos desordenados se encuentran cercanos, o lejanos.

Es bastante adecuado para ordenar listas de tamaño moderado, debido a que su velocidad es aceptable y su codificación es bastante sencilla. Su velocidad depende de la secuencia de valores con los cuales trabaja, ordenándolos. El siguiente ejemplo muestra el proceso de forma gráfica:

Considerando un valor pequeño que está inicialmente almacenado en el final del vector. Usando un ordenamiento  $O(n^2)$  como el ordenamiento de burbuja o el ordenamiento por inserción, tomará aproximadamente  $n$  comparaciones e intercambios para mover este valor hacia el otro extremo del vector.

El Shell sort primero mueve los valores usando tamaños de espacio gigantes, de manera que un valor pequeño se moverá bastantes posiciones hacia su posición final, con sólo unas pocas comparaciones e intercambios.

278787  
278828  
278816  
266145

Carlos Emilio Cortés Martínez  
Fernando Maya Carranza  
Diego A. Lomeli Vazquez  
Ivonne Monserrat Cruz Paz