Unidad 2

2.1)Estructuras de control en Ruby

Temas: Temas: Estructura de control en Ruby: while, each, switch, if, for, ciclos incrementables. Rangos. Tiempo. Dibujar patrones con ciclos. Ciclos anidados.

Estructuras de control

Hasta acá habíamos ejecutado código que imprime texto, operaciones matemáticas y hemos evaluado algunas condiciones o valores de los datos.

En esta unidad revisaremos las estructuras de control, ciclos, iteradores, que permiten ya no solamente evaluar el valor de un objeto, también permitirán generar o desencadenar rutinas, asignaciones o respuestas mientras se cumpla una condición o varias condiciones.

while

While

La estructura es la siguiente:

```
while condición
   hace esto
end
```

016While.rb — C:\Users\hacking\Downloads\ruby — Atom

File Edit View Selection Find Packages Help

Project

- ruby
  - 1
    - 01Puts.rb
    - 02Math.rb
    - 03Var.rb
    - 04Time.rb
    - 05Error.rb
    - 06Hashes.rb
    - 07Hashes.rb
    - 08Math2.rb
    - 09Math2.rb
    - holamundo.rb
  - 2
    - bonus
    - .DS_Store
    - 08Get.rb
    - 09Estructura.rb
    - 010Estructura.rb
    - 011Metodo.rb
    - 012Rescate.rb
    - 013For.rb
    - 014While.rb
    - 015Each.rb
    - 016While.rb
    - tallerSesion2.pdf

holamundo.rb | 03Var.rb | 09Math2.rb | 09Estructura.rb | 016While.rb | 010Estructura.rb

```ruby
titulo1 = "Módulo introducción a Ruby" #cadenas de texto, strings
titulo2 = "2021"
titulo3 = "CHILE"

puts titulo1
puts titulo2
puts titulo3

cuenta = 0




    i = 0
    while  i <= 12 do

    i += 1
    sleep(1)#observar que sucede si no ubicamos este sleep
    puts "#{i} #{"+"}"

    end
```

2\016While.rb* 28:1 LF UTF-8 Ruby GitHub Git (0)

Escribe aquí para buscar

ESP 10:39 07/02/2021

each

El ciclo each permitirá iterar sobre un objeto, un array por ejemplo, permitiendo traer todos los datos de ese array de manera simultánea, especialmente útil por ejemplo a la hora de hacer consultas en BBDD. Permite también hacer una lectura retrógrada con el comando **.reverse_each**

```
arr = ["Mane, Carolina, Claudia, Marco, Sebastián, Eduardo, Freddy"]

arr.each do |i|

   puts i

end


nums = %w[uno dos tres cuatro cinco seis siete ocho]

str = " "

nums.reverse_each { |nums| str += "#{nums} " }

nums.each { |nums| str += "#{nums} " }
```

015Each.rb — C:\Users\hacking\Downloads\ruby — Atom

File Edit View Selection Find Packages Help

Project

- ruby
  - 1
    - 01Puts.rb
    - 02Math.rb
    - 03Var.rb
    - 04Time.rb
    - 05Error.rb
    - 06Hashes.rb
    - 07Hashes.rb
    - 08Math2.rb
    - 09Math2.rb
    - holamundo.rb
  - 2
    - bonus
    - .DS_Store
    - 08Get.rb
    - 09Estructura.rb
    - 010Estructura.rb
    - 011Metodo.rb
    - 012Rescate.rb
    - 013For.rb
    - 014While.rb
    - 015Each.rb
    - 016While.rb
    - tallerSesion2.pdf

holamundo.rb | 03Var.rb | 09Math2.rb | 09Estructura.rb | 016While.rb | 015Each.rb | 01Puts.rb | 010Estructura.rb

```ruby
titulo1 = "Módulo introducción a Ruby"
titulo2 = "2021"
titulo3 = "CHILE"

puts titulo1
puts titulo2
puts titulo3

puts titulo1, titulo2, titulo3

arr = ["Mane, Carolina, Claudia, Marco, Sebastián, Eduardo, Freddy"]

arr.each do |i|
   puts i
end

nums = %w[uno dos tres cuatro cinco seis siete ocho]
str = ""
nums.reverse_each { |nums| str += "#{nums} " }
nums.each { |nums| str += "#{nums} " }


p str #=> "sixth fifth fourth third second first "

```

2\015Each.rb 24:1 LF UTF-8 Ruby GitHub Git (0)

Escribe aquí para buscar

ESP 10:54 07/02/2021

switch case

El switch permite trabajar con casos (case), se pueden ejecutar diferentes respuestas de acuerdo al caso.

Switch

```
case
   when
   else #en caso que no coincida ningún caso

end
```

017Case.rb — C:\Users\hacking\Downloads\ruby — Atom

File Edit View Selection Find Packages Help

Project

ruby
- 1
  - 01Puts.rb
  - 02Math.rb
  - 03Var.rb
  - 04Time.rb
  - 05Error.rb
  - 06Hashes.rb
  - 07Hashes.rb
  - 08Math2.rb
  - 09Math2.rb
  - holamundo.rb
- 2
  - bonus
  - .DS_Store
  - 08Get.rb
  - 09Estructura.rb
  - 010Estructura.rb
  - 011Metodo.rb
  - 012Rescate.rb
  - 013For.rb
  - 014While.rb
  - 015Each.rb
  - 016While.rb
  - 017Case.rb

holamundo.rb | 03Var.rb | 09Math2.rb | 09Estructura.rb | 016While.rb | 017Case.rb | 01Puts.rb | 010Estructura.rb

```ruby
puts titulo1
puts titulo2
puts titulo3

print "Adivina el nombre...nombres con c, m, j y v, ingresa el nombre que tú creas en minúscula: "


nombre = gets.chomp


case nombre


when "carlos"
  puts 'El nombre es Carlos'

when "mario"
  puts 'El nombre es Mario'

when "jorge"
  puts 'El nombre es Jorge'

when "valeria"
  puts 'El nombre es Valeria'

else
  puts "Error!"

end
```

2\017Case.rb 18:14 LF UTF-8 Ruby GitHub Git (0)

Escribe aquí para buscar

ESP 11:16 07/02/2021

for

El ciclo for ejecutará una instrucción mediante la evaluación de una condición

En un lenguaje tradicional el for sería_

```
for (int x = 0, x < 10, x+1)
```

Donde x está inicializada con valor 0, luego se evalúa, x es menor que 0, si es verdad le suma 1 a x, por tanto x ahora vale 2, hará esta operación hasta que x sea menor a 10, luego terminará el proceso. El for en ruby establece rangos, es un poco diferente a lo que estábamos habituados.

018For.rb — C:\Users\hacking\Downloads\ruby — Atom
File Edit View Selection Find Packages Help
Project
03Var.rb
04Time.rb
05Error.rb
06Hashes.rb
07Hashes.rb
08Math2.rb
09Math2.rb
holamundo.rb
2
bonus
.DS_Store
08Get.rb
09Estructura.rb
010Estructura.rb
011Metodo.rb
012Rescate.rb
013For.rb
014While.rb
015Each.rb
016While.rb
017Case.rb
018For.rb
tallerSesion2.pdf
programming-ruby.pdf
rubyinstaller-devkit-2.7.2-1-x64.exe
The Ruby Programming Language - Oreilly.
holamundo.rb
03Var.rb
09Math2.rb
09Estructura.rb
016While.rb
017Case.rb
018For.rb
04Time.rb
010Estructura.rb
1 titulo1 = "Módulo introducción a Ruby"
2 titulo2 = "2021"
3 titulo3 = "CHILE"
4
5 puts titulo1
6 puts titulo2
7 puts titulo3
8
9
10
11 textos = "Ingresa un número y presiona enter"
12 puts textos
13
14 numero1 = gets.chomp
15
16 textos2 = "Ingresa otro número y presiona enter"
17 puts textos2
18
19 numero2 = gets.chomp
20
21 entero1 = numero1.to_i
22 entero2 = numero2.to_i
23
24
25
26 for i in(entero1..entero2)
27 puts i
28 end
29
2\018For.rb* 6:13
LF UTF-8 Ruby GitHub Git (0)
Escribe aquí para buscar
ESP
11:34
07/02/2021

Tiempo

Ruby también permite trabajar con tiempo, especialmente útil para automatizar rutinas, ciclos o iteraciones.

04Time.rb — C:\Users\hacking\Downloads\ruby — Atom
File Edit View Selection Find Packages Help
Project
ruby
1
01Puts.rb
02Math.rb
03Var.rb
04Time.rb
05Error.rb
06Hashes.rb
07Hashes.rb
08Math2.rb
09Math2.rb
groupby.rb
hashes3.rb
hashes4.rb
holamundo.rb
iteracionhash.rb
mergehash.rb
2
bonus
.DS_Store
08Get.rb
09Estructura.rb
010Estructura.rb
011Metodo.rb
012Rescate.rb
pais.json
020Metodo.rb
04Time.rb
1 titulo1 = "Módulo introducción a Ruby"
2 titulo2 = "2021"
3 titulo3 = "CHILE"
4
5
6 puts titulo1
7 puts titulo2
8 puts titulo3
9
10
11
12 tiempo = Time.new
13
14
15 hora = tiempo.hour
16 minuto = tiempo.min
17 segundo = tiempo.sec
18 hora_completa = "#{hora}:#{minuto}:#{segundo}"
19
20 puts hora_completa
21
22 ano = tiempo.year
23 mes = tiempo.month
24 dia = tiempo.day
25 fecha_completa = "#{dia}/#{mes}/#{ano}"
26
27 puts fecha_completa
28
1\04Time.rb 11:1
LF UTF-8 Ruby GitHub Git (0)
Escribe aquí para buscar
0:26
15/02/2021

Ciclos anidados

Como cualquier otro lenguaje de programación es posible anidar ciclos, en el siguiente ejemplo revisaremos cómo anidar un while

Project — C:\Users\hacking\Downloads\ruby — Atom

File Edit View Selection Find Packages Help

Project

- 03Var.rb
- 04Time.rb
- 05Error.rb
- 06Hashes.rb
- 07Hashes.rb
- 08Math2.rb
- 09Math2.rb
- holamundo.rb
- 2
  - bonus
  - .DS_Store
  - 08Get.rb
  - 09Estructura.rb
  - 010Estructura.rb
  - 011Metodo.rb
  - 012Rescate.rb
  - 013For.rb
  - 014While.rb
  - 015Each.rb
  - 016While.rb
  - 017Case.rb
  - 018For.rb
  - tallerSesion2.pdf
- programming-ruby.pdf
- rubyinstaller-devkit-2.7.2-1-x64.exe
- The Ruby Programming Language - Oreilly

holamundo.rb | 03Var.rb | 09Math2.rb | 09Estructura.rb | 016While.rb | 014While.rb | 017Case.rb | 018For.rb | 010Estructura.rb

```ruby
titulo1 = "Módulo introducción a Ruby" #cadenas de texto, strings
titulo2 = "2021"
titulo3 = "CHILE"

puts titulo1
puts titulo2
puts titulo3

cuenta = 0



  while cuenta <= 1
    i = 0
    while  i <= 12 do

    i += 1
    sleep(1)#observar que sucede si no ubicamos este sleep
    puts "#{i} #{"+"}"

  end

  while  i >= 1 do

    i -= 1
    sleep(1)#observar que sucede si no ubicamos este sleep
    puts "#{i} #{"-"}"

  end

```

2\014While.rb 1:1 LF UTF-8 Ruby GitHub Git (0)

Escribe aquí para buscar

ESP 11:41 07/02/2021

Material complementario de la unidad

Link a video relacionado

1. https://www.youtube.com/watch?v=qDdbArXJkIg

Link a lectura complementaria

1. https://docs.microsoft.com/es-es/dotnet/visual-basic/programming-guide/language-features/control-flow/loop-structures

Link a investigación relacionada

1.https://www.rubyguides.com/ruby-tutorial/loops/