



Flujo

Sesión Conceptual 2





Inicio

{desafío}
latam_



15 minutos

- Construir un programa en Ruby a partir de un diagrama de flujo, utilizando operadores lógicos.
- Simplificar problemas con condiciones anidadas en Ruby.
- Optimizar un programa en ruby a partir del análisis de condiciones de borde.

Objetivo



Desarrollo

{desafío}
latam_



150 minutos

/* Introducción al manejo de flujo */

La instrucción IF

Ruby -como todo lenguaje de programación- tiene instrucciones para implementar condiciones.

- La más utilizada es la instrucción if.

```
if condition
  # código que se ejecutará SÓLO si se cumple la condición
end
```

Importante

- La instrucción end es muy importante, por cada if tiene que haber un end.
- Todo lo que está dentro del if, sucederá sólo si se cumple la condición.
- El código dentro del if debe estar indentado para poder reconocer de forma sencilla y visualmente dentro del código
- Dónde empieza y termina y que código se ejecuta dentro de la condición.

Ejercicio

Contexto:

Si ejecutamos el programa e introducimos un valor mayor o igual a 18 veremos el mensaje "Eres mayor de edad" en caso contrario, no veremos ningún mensaje.

En este ejercicio comparamos utilizando el operador `>=`, pero existen varios operadores que nos permiten comparar.



/* Operaciones de comparación */

Operadores de comparación en enteros

Operador	Nombre	Ejemplo => Resultado
==	Igual a	2 == 2 => true
!=	Distinto a	2 != 2 => false
>	Mayor a	3 > 4 => false
>=	Mayor o igual a	3 >= 3 => true
<	Menor a	4 < 3 => false
<=	Menor o igual a	3 <= 4 => true

Probando las comparaciones en IRB

```
Gonzalo-2:examples gonzalosanchez$ ruby ex1.rb
¿Qué edad tienes?
19
Eres mayor de edad
Gonzalo-2:examples gonzalosanchez$ ruby ex1.rb
¿Qué edad tienes?
5
Gonzalo-2:examples gonzalosanchez$
```

Realicemos una prueba en IRB, donde el usuario ingrese los valores y veamos si el primero es mayor que el segundo.

Expresiones más complejas

A veces deseamos operar con los datos que tenemos antes de comparar. Podemos combinar los operadores de comparación con los métodos que deseemos.

```
a = 3
b = 4
puts 2 * a > b + 4 # true
```

Asignación vs Comparación

Un error muy frecuente de principiante es intentar comparar utilizando un signo igual en lugar de utilizar dos.

Con comparación:

```
a = 3  
a == 2 # false
```

Con asignación:

```
a = 2  
a = 3 # 3
```

Operadores de comparación en strings

Aunque en la tabla sólo hayamos mostrado números, podemos comparar dos objetos utilizando un operador de comparación:

```
'texto1' == 'texto2' # false
```

¿Puede un texto ser mayor que otro?

En este caso la comparación es por orden alfabético, las letras que aparecen primero en el alfabeto son menores que las que aparecen después. Para entenderlo de forma sencilla: Cuando comparamos dos palabras es menor la que aparecería antes en un diccionario.

/* Profundizando en flujo */

Abordemos el problema desde un ejemplo

Utilizaremos el mismo ejemplo manejando el flujo cuando ambos números ingresados sean iguales.

Antes:	Ahora:
<p>caso1: $A > B \Rightarrow$ Decimos que A es mayor.</p> <p>caso2: $A < B \Rightarrow$ Decimos que A es menor</p>	<p>caso1: $A > B \Rightarrow$ Decimos que A es mayor.</p> <p>caso2: $A == B \Rightarrow$ Decimos que A y B son iguales.</p> <p>caso3: $A < B \Rightarrow$ Decimos que A es menor que B</p>

El mayor de dos números: Escenario donde se ingresan dos números iguales

Ante cualquier problema de este tipo lo mejor es ir paso a paso y analizar el problema.

1. Si el primero es mayor -> Mostramos mensaje.
2. Si NO es mayor -> Tenemos dos opciones:
 - El primer valor es menor.
 - Ambos son iguales

Transcribiendo el diagrama

Para escribir este código crearemos el archivo `mayor_menor_o_igual.rb` y dentro agregaremos nuestro código.

```
puts 'Ingrese valor1:'
valor1 = gets.to_i #asignación

puts 'Ingrese valor2:'
valor2 = gets.to_i #asignación

if valor1 > valor2 #comparación
  puts "valor1 #{valor1} es mayor"
else
  if valor1 == valor2 #comparación
    puts 'ambos valores son iguales'
  else
    puts "valor2 #{valor2} es mayor"
  end
end
end
```

Un código siempre debe ser probado en todos los casos, por lo que probaremos el programa 3 veces, ingresando los siguientes valores:

- 2, 1.
- 2, 2.
- 2, 3.

Existe otra solución sin tener que agregar un `if` dentro de otro.

La instrucción ELSIF

La instrucción `elsif` nos permite capturar el flujo y volver a realizar una evaluación condicional cuando no se cumplió una evaluación previa. Con esta instrucción podemos reescribir el algoritmo del ejemplo anterior, pero conservaremos el archivo y crearemos uno nuevo, llamado `mayor_menor_o_igual2.rb`.

```
puts 'Ingrese valor1:'
valor1 = gets.chomp.to_i

puts 'Ingrese valor2:'
valor2 = gets.chomp.to_i

if valor1 > valor2
  puts "valor1 #{valor1} es mayor"
elsif valor1 == valor2
  puts 'son iguales'
else
  puts "valor1 #{valor2} es menor"
end
```

Clasificación según rangos

Es normal estar en situaciones donde nos pidan clasificar algún elemento según un rango.

Por ejemplo, supongamos que tenemos una palabra y queremos clasificarla en corta, mediana y larga:

- 4 letras o menos será corta.
- 5 a 10 será mediana.
- Más de 10 será larga.

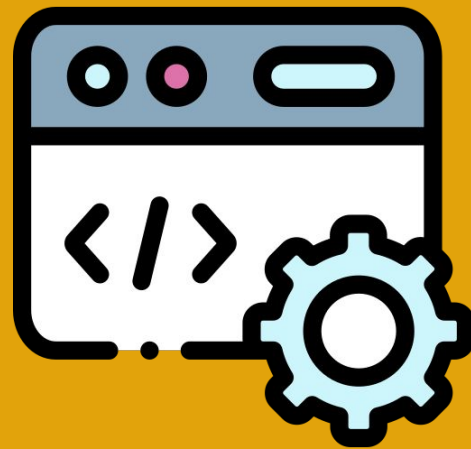
```
puts 'Ingresa una palabra'
palabra = gets.chomp
largo = palabra.size

if largo <= 4
  puts 'Pequeña'
elsif largo < 10
  puts 'Mediana'
else
  puts 'Larga'
end
```

Ejercicio

Contexto:

Modifica el código anterior para poder distinguir palabras muy largas, cuando tengan 15 o más caracteres.



/* Condiciones de borde */

Analizando una condición de borde

```
puts "¿Qué edad tienes?"  
edad = gets.chomp.to_i  
  
if edad >= 18  
  puts "Eres mayor de edad"  
else  
  puts "Eres menor de edad"  
end
```

En este ejemplo, el punto que define una rama de la otra es el 18. A este punto se le suele llamar punto crítico, mientras que los bordes son los números que "bordean" el 18, es decir, analizar los bordes corresponde a probar con los valores 17, 18 y 19.

/* Operadores lógicos */

Operadores lógicos

Operador	Nombre	Ejemplo	Resultado
&&	y (and)	false && true	Devuelve true si ambos operandos son true, en este ejemplo se devuelve false.
	o (or)	false true	Devuelve true si al menos uno de los operandos es true, en este ejemplo devuelve true.
!	no (not)	!false	Devuelve lo opuesto al resultado de la evaluación, en este ejemplo devuelve true.

Identities

'Igual' es lo mismo que 'no distinto': Negar algo dos veces es afirmarlo (en español, no siempre es así; en programación, sí).

Mayor y no menor igual: Un caso similar es la comparación $a > 18$. Decir que a no es mayor a 18, es decir que es menor o igual a 18, (debemos incluir el 18 al negar).

Unless: Para ayudarnos a escribir las condiciones siempre en positivo, existe una instrucción que es el antónimo del if, está se llama unless.

/* Simplificando IFs anidados */

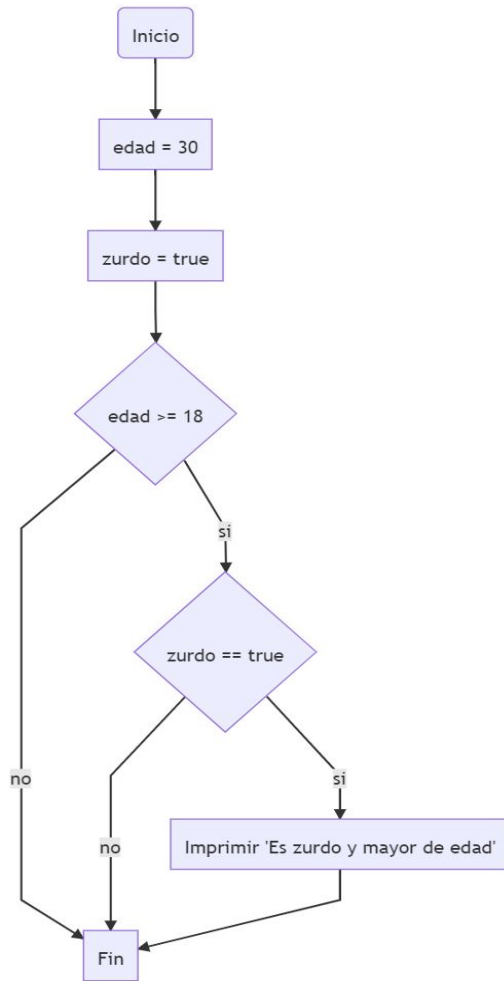
Identificando ifs anidados

```
edad = 30
zurdo = true

if edad >= 18
  if zurdo == true
    puts 'Es zurdo y mayor de edad'
  end
end
```

En el ejemplo anterior vemos un if dentro de otro if: a esto se le llama tener if anidados.

El código escrito muestra el texto sólo si se cumplen las dos condiciones.



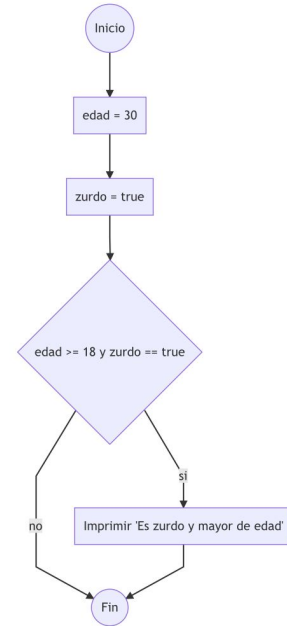
Operadores lógicos

Podemos simplificar el código para evaluar ambas condiciones en una misma instrucción if.

- ¿Qué operador podemos utilizar si necesitamos que ambas condiciones sean verdaderas?

```
edad = 30
zurdo = true

if edad >= 18 && zurdo == true
  puts 'Es mayor de edad y zurdo'
end
```



Ejercicio de integración

Se busca crear un programa que solicite al usuario ingresar tres números. El programa debe determinar el mayor de ellos. Se asume que los números ingresados serán distintos.



/* Simplificando el flujo */

Variantes de IF

If en una línea (inline)

En Ruby, también es posible utilizar versiones cortas de la instrucción if y unless de la siguiente forma: action if condition.

Operador ternario

El operador ternario es una variante de if que permite operar en base a dos caminos en condiciones simples.

La lógica es la siguiente:

```
si_es_verdadero    ?    entonces_esto    :  
sino_esto
```

Refactorizar

Podemos refactorizar las comparaciones en los condicionales que son innecesarias.

¿Cómo?: Recordemos que la instrucción `if` espera que el resultado se evalúe como `true` o `false`. Por lo tanto:

```
mayor_de_edad = true

if mayor_de_edad == true
  puts "Mayor de edad"
end
```

Es igual:

```
mayor_de_edad = true

if mayor_de_edad
  puts "Mayor de edad"
end
```


Análisis léxico

Existe un programa encargado de traducir cada una de nuestras instrucciones a un lenguaje que nuestro computador pueda entender.

Conocer los procesos que realiza Ruby al leer un programa nos permitirá entender el por qué son necesarias estas reglas y nos ayudará a memorizarlas.

- **¿Qué sucede cuando ingresamos al terminal y escribimos ruby mi_programa.rb?**
 - Ruby lee y procesa nuestro código.

El proceso

Etapa 1: Tokenización

Ruby, al leer un código, lo primero que realiza es una tokenización. Esto consiste en separar cada palabra o símbolo en unidades llamadas tokens.

Etapa 2: Lexing

A través de un proceso llamado lexing se clasifican los tokens. Cada token es identificado según reglas.

Etapa 3: Parsing

Consiste en generar un árbol llamado árbol de sintaxis abstracta. Esto corresponde a una forma de representar el código que le permite -a Ruby- saber en qué orden ejecutar las operaciones, o descubrir si todo paréntesis abierto se cierra.

Reglas Básicas

Ahora estudiaremos las reglas básicas de Ruby a partir de los tipos de tokens.

Mencionamos anteriormente que en una etapa los tokens son clasificados. Esta clasificación los separa en:

- Identificadores.
- Literales.
- Comentarios.
- Palabras reservadas.



Quiz

{desafío}
latam_



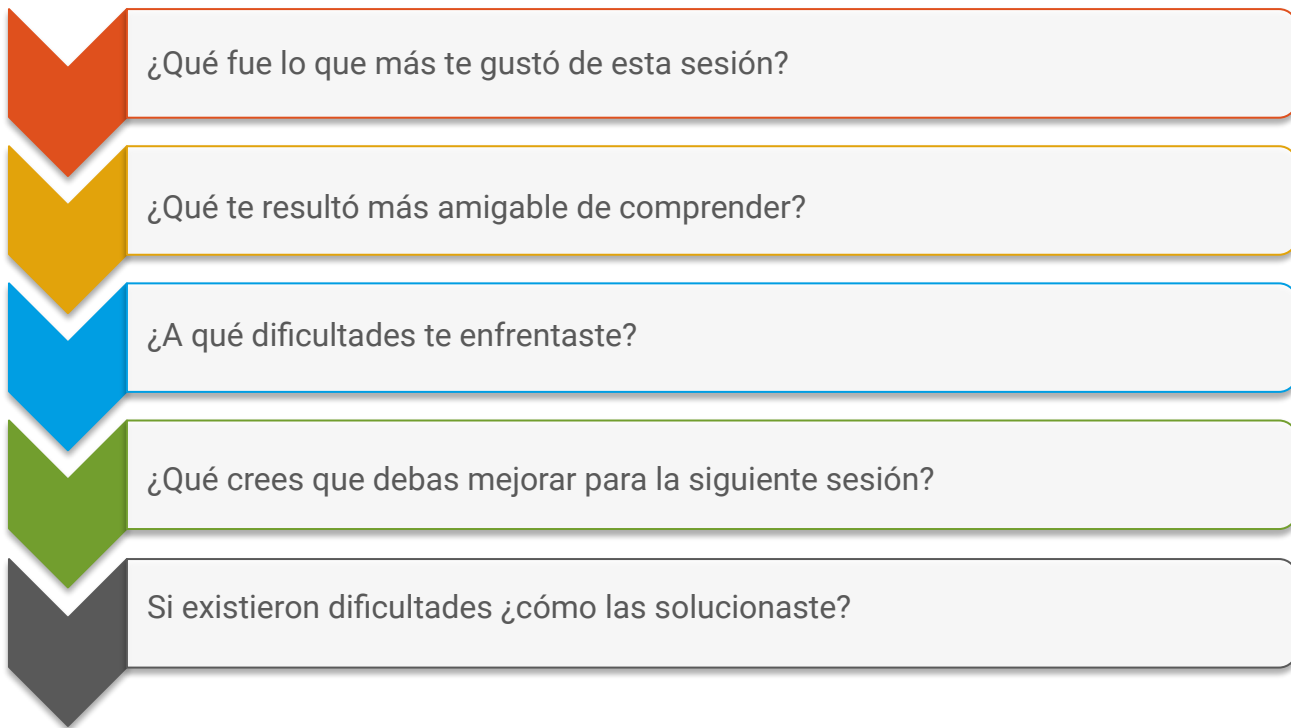


Cierre

{desafío}
latam_



15 minutos



¿Qué fue lo que más te gustó de esta sesión?

¿Qué te resultó más amigable de comprender?

¿A qué dificultades te enfrentaste?

¿Qué crees que debas mejorar para la siguiente sesión?

Si existieron dificultades ¿cómo las solucionaste?



*Academia de
talentos digitales*

www.desafiolatam.com



/DesafioLatam



/DesafioLatam



/DesafioLatam



/DesafioLatam