Introducción a manejo de flujo

¡Comencemos!

¿Qué aprenderás?

- Crear algoritmos que tomen decisiones en base al valor de una variable.
- Utilizar la instrucción if para tomar decisiones.
- Comparar números enteros.
- Comparar strings.
- Diferenciar comparación de asignación.
- Manejar flujos en situaciones de un caso o en caso contrario.
- Utilizar la instrucción else.

Introducción

En algunas situaciones necesitaremos crear algoritmos que tomen una decisión en base a una condición, por ejemplo, si un valor es mayor a una cantidad se hace el paso 1 pero si es menor no se hace.

Cuando el algoritmo tiene caminos a seguir entonces empezamos a hablar de flujo.

¡Vamos con todo!

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
```

Lo anterior se lee como: "Si se cumple la condición, entonces ejecuta el código".

Ejemplo de if

Analicemos el siguiente ejemplo: En muchos países de Latinoamérica, la mayoría de edad se cumple a los 18 años. Crearemos un programa llamado `mayor_edad.rb` que pregunte la edad al usuario. Si la edad es mayor o igual a 18 entonces le diremos que es mayor de edad.

Diagrama de flujo

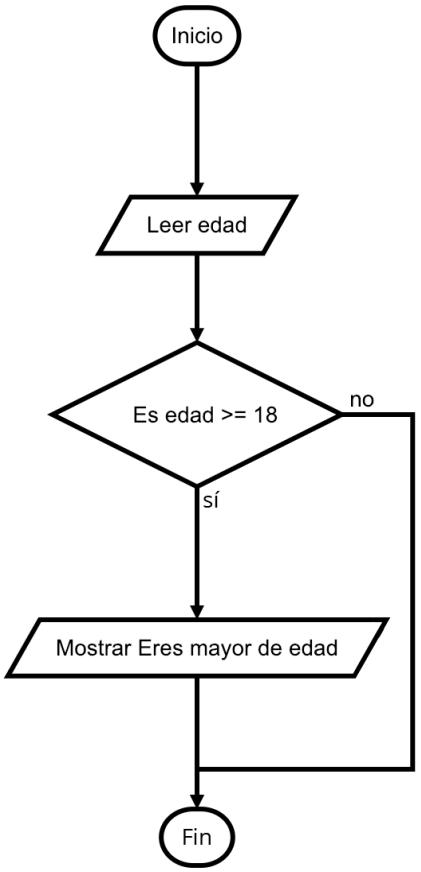

Imagen 1. Ejemplo de diagrama de flujo.
Fuente: Desafío Latam.

Siempre es bueno ver primero el diagrama de flujo

Implementemos el código

Dentro de nuestra carpeta de trabajo crearemos el programa mayor_de_edad.rb con nuestro editor favorito, donde agregaremos:

```
puts "¿Qué edad tienes?"
edad = gets.chomp.to_i

if edad >= 18
   puts "Eres mayor de edad"
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.

Probando el programa

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, estos los estudiaremos en el siguiente capítulo.

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

Tabla 1. Operadores de comparación.
Fuente: Desafío Latam.

Probando las comparaciones en IRB

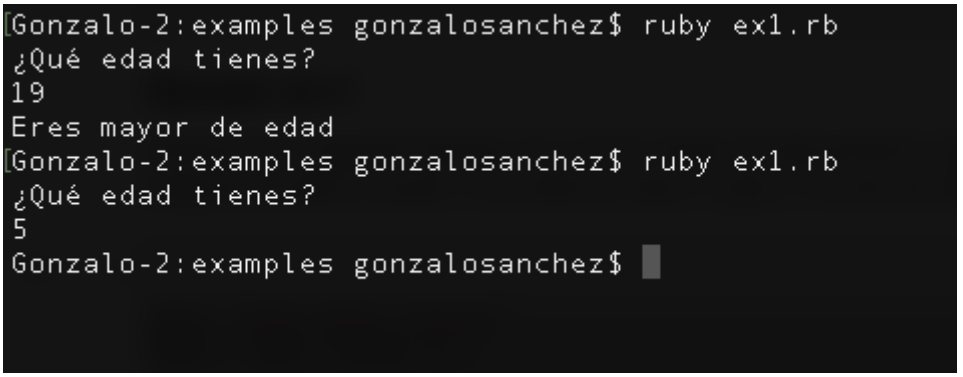

Imagen 2. Pruebas de comparación.
Fuente: Desafío Latam.

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

```
a = 3
b = 4
puts a > b # false
```

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
```

En este caso es importante notar que el operador de comparación tiene menor precedencia que cualquier operador aritmético, por lo que primero se evaluarán las expresiones a cada lado del comparador, y luego se compararán los resultados

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?

```
'a' < 'b' # true
```

Imagen 3. Orden de palabras.
Fuente: Desafío Latam.

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 que la que aparecería antes en un diccionario.

En la imagen 3 vemos que la palabra recycle es menor que la palabra red porque la letra 'c' es menor que 'd'.

En caso contrario

¿Cómo podemos modificar nuestro programa para que muestre un mensaje cuando el usuario sea menor edad y otro mensaje cuando sea mayor de edad?

Una muy buena práctica es la de realizar un diagrama de flujo antes de comenzar a programar. Esto ayuda a abstraerse del código y pensar en los pasos críticos.

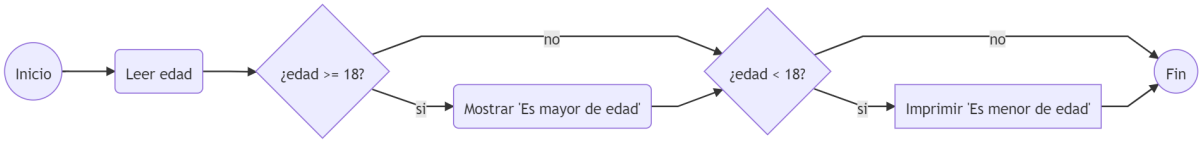

Imagen 4. Ejemplo de Diagrama de flujo, comprobar mayoría de edad.
Fuente: Desafío Latam.

Transcribimos nuestro diagrama de flujo a código Ruby:

```ruby
puts "¿Qué edad tienes?"
edad = gets.chomp.to_i

if edad >= 18
   puts "Eres mayor de edad"
end

if edad < 18
 puts "Eres menor de edad"
end
```

Una pregunta interesante que nuestro diagrama refleja de manera implícita es: ¿Puede el usuario ser mayor y menor de edad a la vez?

La respuesta es, evidentemente, no. Este tipo de situaciones se puede modelar mejor de la siguiente forma.

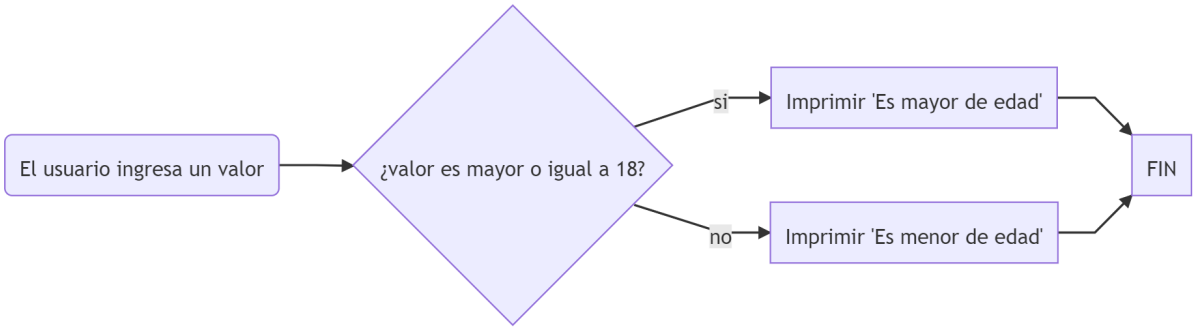

Imagen 5. Diagrama simplificado, comprobar mayoría de edad.
Fuente: Desafío Latam.

Para implementarlo tenemos que introducir una nueva instrucción: else

La instrucción ELSE

La instrucción `else` se utiliza junto con `if` para seguir el flujo de código en cualquier caso donde no se cumpla la condición.

```
if condition
   # código que se ejecutará SÓLO SI se cumple la condición
else
   # código que se ejecutará si NO se cumple la condición
end
```

Implementemos el código de mayor de edad con if y else

```
puts "¿Qué edad tienes?"
edad = gets.to_i

if edad >= 18
   puts "Eres mayor de edad"
else
   puts "Eres menor de edad"
end
```

Ejercicio resuelto

Crear un programa mayor_de_2.rb en nuestra carpeta de trabajo donde el usuario deba ingresar dos números. El programa debe imprimir el mayor de ambos números.

```
puts "Ingrese valor1"
valor1 = gets.to_i
puts "Ingrese valor2"
valor2 = gets.to_i

if valor1 >= valor2
   puts "valor1 #{valor1} es mayor"
else
   puts "valor2 #{valor2} es mayor"
end
```

Ejercicios

1. Crear el programa password.rb en la carpeta de trabajo donde el usuario debe ingresar un password en la plataforma, si el password tiene menos de 6 letras se debe mostrar un aviso de alerta que el password es muy corto.

2. Crear el programa password2.rb donde el usuario debe ingresar un password, si el password es 12345 se debe informar que el password es correcto pero en caso contrario se debe mostrar que el password es incorrecto.

Resumen de lo aprendido

- En este capítulo aprendimos a utilizar las instrucciones if y else, las cuales nos permiten manejar el flujo de un programa.
- Existen operadores de comparación que nos permiten comparar si una expresión es verdad o mentira.
- Podemos utilizar los operadores de comparación dentro de la condición de los if.
- Debemos tener cuidado de no confundir la asignación de la comparación. Es decir: = es distinto de ==.
- La instrucción end es 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, termina y qué código se ejecuta dentro de la condición