Ciclos

¡Comencemos!

¿Qué aprenderás?

- Conocer los ciclos y sus posibles aplicaciones en la programación.
- Leer y transcribir diagramas de flujo con iteraciones a código Ruby.
- Validar una entrada de datos.
- Crear un menú de opciones

Introducción

Los ciclos son instrucciones que nos permiten repetir la ejecución de una o más instrucciones.

```
Mientras se cumple una condición:
   Instrucción 1
   Instrucción 2
   Instrucción 3
```

Repetir instrucciones es la clave para crear programas avanzados.

¡Vamos con todo!

Usos de ciclos

Los posibles usos de ciclos en algoritmos son infinitos, nos permiten recorrer colecciones de datos o espacios de búsqueda.

Parte importante de aprender a programar es entender cómo utilizarlos correctamente y desarrollar las habilidades lógicas para entender cuándo utilizarlos.

En esta unidad estudiaremos distintos problemas que se resuelven con ciclos y que, si bien no siempre son utilizados en la industria, nos ayudarán a desarrollar las habilidades lógicas que necesitamos para ser buenos programadores.

WHILE

La instrucción while nos permite ejecutar una o más operaciones mientras se cumpla una condición. Su sintaxis es la siguiente:

```
while(condition)
 # Código que se ejecuta
end
```

While paso a paso

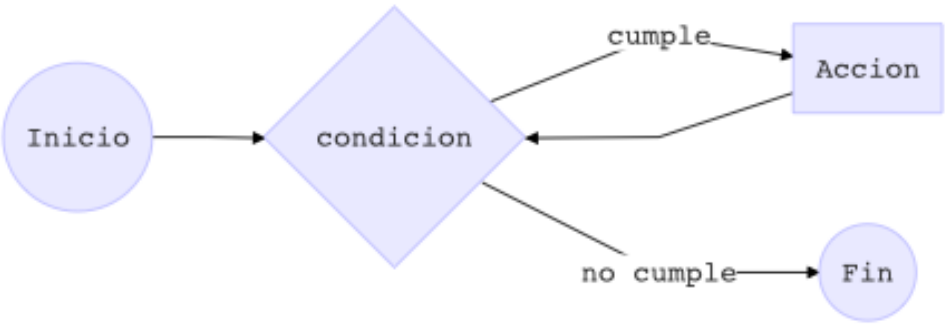

Imagen 1. While.
Fuente: Desafío Latam.

1. Se evalúa la condición; si es true, ingresa al ciclo.

2. Se ejecutan, secuencialmente, las instrucciones definidas dentro del ciclo.

3. Una vez ejecutadas todas las instrucciones se vuelve a evaluar la condición:

    - Si se evalúa como true: vuelve a repetir.
    - Si se evalúa como false: sale del ciclo.

Salida del ciclo

Un algoritmo es una secuencia FINITA de pasos para resolver un problema. En algún momento alguna de las instrucciones ejecutadas al interior del ciclo debe lograr que la condición deje de cumplirse.

Validación de entrada de datos utilizando while

Un ejemplo común para comenzar a estudiar los ciclos es validar la entrada de un dato, es decir, que este dato cumpla un criterio. Podemos, por ejemplo, validar que el usuario ingrese un número entre 1 y 10:

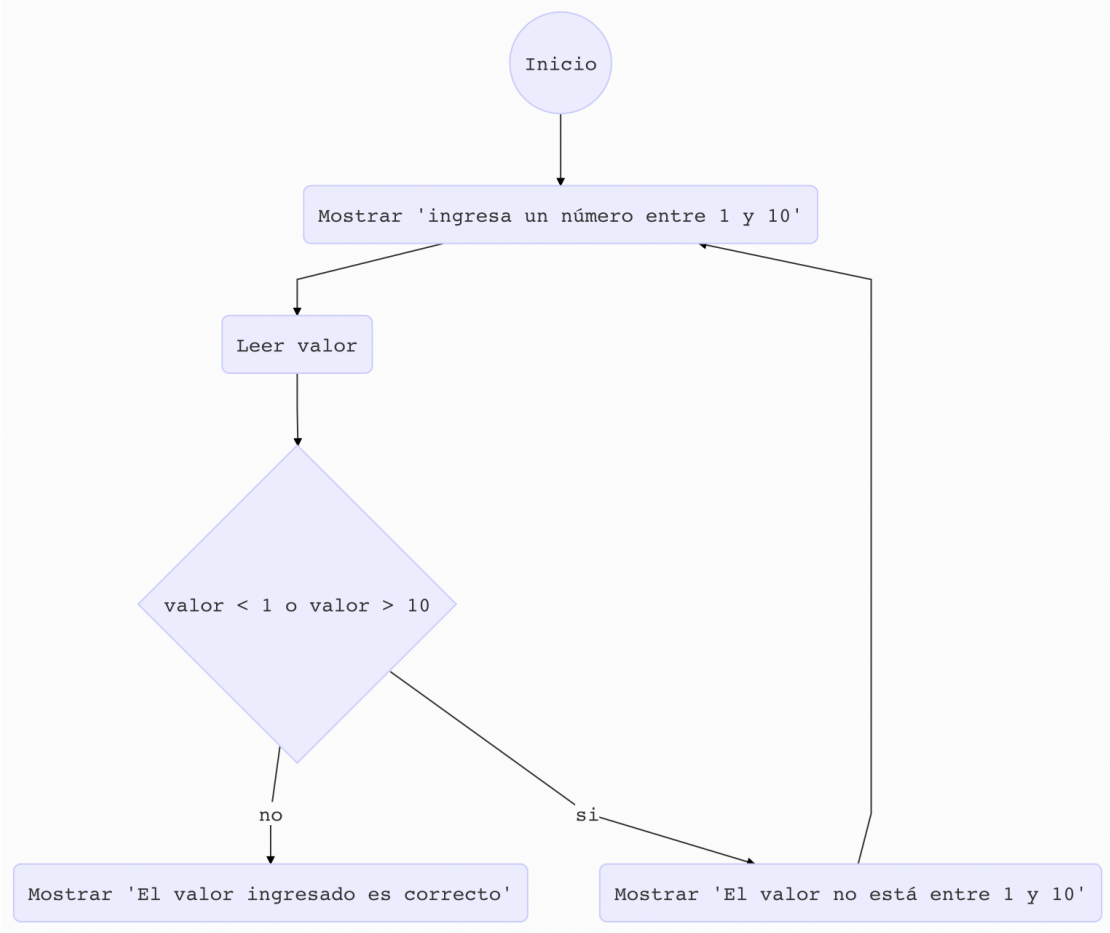

Imagen 2. Validación de entrada.
Fuente: Desafío Latam.

Para escribir este código crearemos el programa `primer_ciclo.rb`.

```
puts 'Ingresa un número entre 1 y 10: '
num = gets.to_i

while num < 1 || num > 10
   puts 'El número ingresado no está entre 1 y 10'
   puts 'Ingresa un número entre 1 y 10: '
   num = gets.to_i
end

puts "El número ingresado fue #{num}."
```

¿Por qué es necesario declarar dos veces el ingreso de datos por parte del usuario?

Si no solicitamos al usuario el ingreso del dato antes del ciclo entonces, cuando el flujo llegue a la evaluación en el ciclo while, la variable num no estará definida, no ingresaremos nunca al ciclo y obtendremos un error.

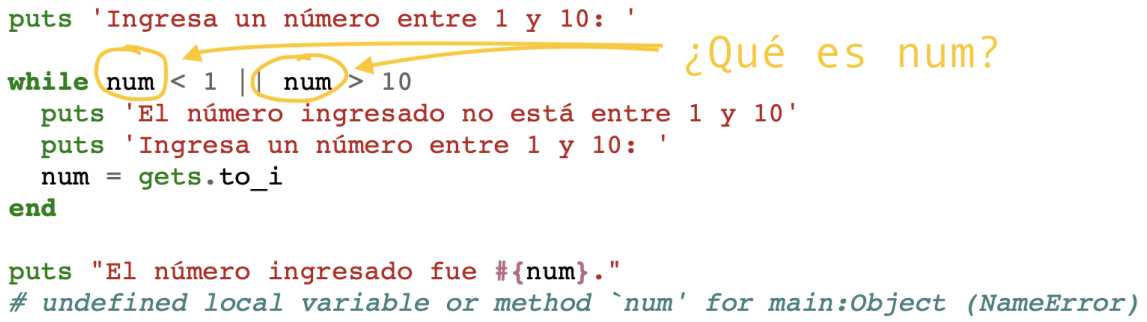

Imagen 3. Dos ingresos.
Fuente: Desafío Latam.

Until

Existe una instrucción que es la inversa de while.

En decir, `while i < 0` es lo mismo que `until i >= 0.`

`while` es mucho más utilizado que `until`, pero es bueno conocerlo por si lo encontramos en algún ejemplo.

Aviso sobre la instrucción gets

Durante el desarrollo de programas en general no utilizaremos gets. Al crear programas utilizaremos argumentos con ARGV o valores al azar con rand, o traeremos los datos desde archivos.

El uso de esta instrucción gets, bloquea el programa hasta que el usuario ingresa un valor, esto complica el uso y evaluación de scripts. De todas formas, podemos agradecer que gets nos ayudó a realizar ejercicios simples con ciclos.

Ejercicio: validación de password

- Crear un diagrama de flujo con ciclos.
- Crear un programa llamado `password_validation.rb` que valide la contraseña de un usuario.

Podemos utilizar la misma idea de validación aprendida para impedir al usuario entrar hasta que ingrese la contraseña "password". Recuerda hacer el diagrama de flujo antes del código. Puedes utilizar el del capítulo anterior como base

Solución

```
puts 'Ingrese su contraseña:'
password = gets.chomp

while password != 'password'
 puts 'La contraseña es incorrecta'
 puts 'Ingrese su contraseña:'
 password = gets.chomp
end

puts "La contraseña ingresada es correcta!"
```

Ejercicio menú de opciones

Podemos implementar de forma sencilla un menú de opciones para el usuario.

La lógica es similar a la de la validación de entrada.

- Se muestra un texto con opciones.
- El usuario tiene que ingresar una opción válida -> validación de entrada.
- Si el usuario ingresa la opción 1 mostramos un texto.
- Si el usuario ingresa la opción 2 mostramos otro texto.
- Si el usuario ingresa la opción "salir" terminamos el programa.

Desarrolla el diagrama de flujo de la solución y luego implementa el algoritmo. En el próximo capítulo revisaremos la solución.

Diagrama de flujo del Menú

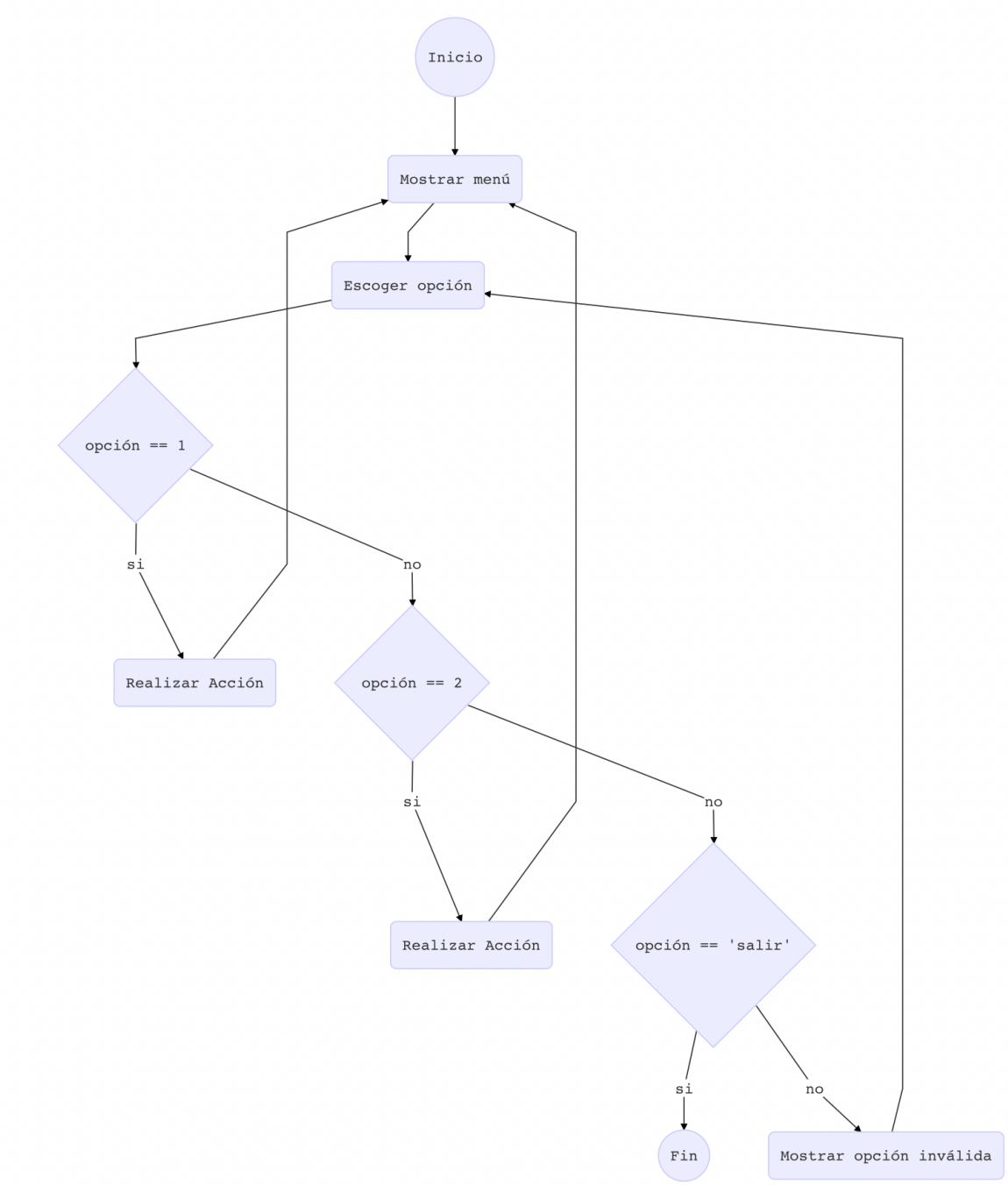

Imagen 4. Diagrama Flujo del Menú.
Fuente: Desafío Latam.

El código para un menú

```
opcion_menu = 'cualquier valor'
while opcion_menu != 'salir' && opcion_menu != 'Salir'
 # Mostrar menú
 puts 'Escoge una opción:'
 puts '-----------------'
 puts '1 - Acción 1'
 puts '2 - Acción 2'
 puts 'Escribe "salir" para terminar el programa'

 puts 'Ingrese una opción:'
 opcion_menu = gets.chomp

 if opcion_menu == '1'
     puts 'Realizando acción 1...'
 elsif opcion_menu == '2'
     puts 'Realizando acción 2...'
 elsif opcion_menu == 'salir' || opcion_menu == 'Salir'
     puts 'Saliendo...'
 else
     puts 'Opción inválida'
 end
end
```