Estructura Algorítmica Repetitiva

Contenido

[Resumen 3](#_Toc169892041)

[Estructura Algorítmica Repetitiva 4](#_Toc169892042)

[Estructura algorítmica Mientras 4](#_Toc169892043)

[Ejercicio 4](#_Toc169892044)

[Errores comunes 5](#_Toc169892045)

[Resumen 6](#_Toc169892046)

[Estructura algorítmica Repetir 6](#_Toc169892047)

[Ejercicio 6](#_Toc169892048)

[Errores comunes 7](#_Toc169892049)

[Resumen 7](#_Toc169892050)

[Estructura algorítmica repetitiva Para(for) 7](#_Toc169892051)

[Ejercicio 7](#_Toc169892052)

[Errores comunes 8](#_Toc169892053)

[Resumen 9](#_Toc169892054)

Resumen

¡Bienvenidos a esta nueva clase donde trataremos las Estructuras **Algorítmicas Repetitivas**! En esta clase, nos enfocaremos en tres tipos principales de estructuras repetitivas:

* **Estructura algorítmica repetitiva Mientras (while)**: Perfecto para situaciones en las que continuamos ejecutando un bloque de código mientras se cumpla una condición.
* **Estructura algorítmica repetitiva Repetir (do-while)**: Útil cuando necesitamos que un bloque de código se ejecute al menos una vez antes de verificar una condición.
* **Estructura algorítmica repetitiva Para (for)**: Ideal para cuando conocemos de antemano cuántas veces necesitamos repetir una tarea.

Estructura Algorítmica Repetitiva

Las estructuras algorítmicas repetitivas son herramientas esenciales en la programación que nos permiten ejecutar un bloque de código múltiples veces. Esto es increíblemente útil cuando necesitamos realizar tareas repetitivas de manera eficiente y efectiva. Los bucles nos ayudan a automatizar procesos, manejar grandes cantidades de datos, y resolver problemas de manera más elegante y simplificada.

## Estructura algorítmica Mientras

Es un bucle de control que permite repetir un bloque de código mientras una condición específica sea verdadera. Se utiliza cuando no se sabe de antemano cuántas veces se necesita repetir la operación, sino que depende de una condición que se evalúa antes de cada iteración.

**Importancia**: Son esenciales para realizar tareas repetitivas basadas en condiciones dinámicas, como procesar entradas de usuario, leer datos hasta el final de un archivo, o realizar cálculos hasta alcanzar una precisión deseada.

**Un ejemplo del mundo real**: Continuar llenando una jarra con agua mientras no esté llena.

### Ejercicio

**Sintaxis:**

1. **`Mientras`:** Palabra clave que indica el inicio del bucle.
2. **`condicion`:** Expresión lógica que se evalúa antes de cada iteración. Si es verdadera, el bucle continúa; si es falsa, el bucle termina.
3. **`Hacer`:** Palabra clave que indica el inicio del bloque de código que se repetirá.
4. **`FinMientras`:** Palabra clave que indica el fin del bucle.

**Imprimir un numero del 1 al 10**

i <- 1

Mientras i <= 10 Hacer

Escribir i

i <- i + 1

FinMientras

**Solución:** Se inicializa i a 1. Mientras i sea menor o igual a 10, se imprime i y luego se incrementa i en 1.

### Errores comunes

**Bucles Infinitos**: Ocurren cuando la condición del bucle nunca se vuelve falsa.

**Ejemplo de bucle infinito**:

i <- 1

Mientras i <= 10 Hacer

Escribir i

// i no se incrementa, por lo que la condición siempre es verdadera

FinMientras

**Condiciones Mal Definidas**: Verifica que la condición del bucle sea correcta y relevante para el problema que estás resolviendo.

x <- 10

Mientras x = 10 Hacer

Escribir "Esto es un bucle infinito"

FinMientras

### Resumen

Un bucle Mientras repite un bloque de código mientras una condición específica sea verdadera.

Es crucial asegurarse de que la condición eventualmente se vuelva falsa para evitar bucles infinitos.

## Estructura algorítmica Repetir

Un bucle Repetir ejecuta un bloque de código repetidamente hasta que una condición específica se vuelve verdadera. A diferencia del bucle Mientras, la condición se evalúa al final de cada iteración, lo que garantiza que el bloque de código se ejecute al menos una vez.

**Importancia**: Los bucles Repetir son útiles cuando queremos asegurarnos de que una acción se realice al menos una vez antes de evaluar una condición, como en la validación de entradas de usuario.

**Ejemplo del mundo real**: Seguir jugando un juego hasta que el jugador decida salir.

### Ejercicio

**Sintaxis:**

* **`Repetir`:** Palabra clave que indica el inicio del bucle.
* **`Hasta Que`:** Palabra clave que indica el final del bucle y evalúa la condición.
* **`condicion`:** Expresión lógica que se evalúa al final de cada iteración. Si es verdadera, el bucle termina; si es falsa, el bucle se repite.

**Pedir una contraseña hasta que sea correcta**:

Repetir

Escribir "Introduce la contraseña:"

Leer contraseña

Hasta Que contraseña = "1234"

### Errores comunes

**Condición Nunca Verdadera**: Asegúrate de que la condición eventualmente se vuelva verdadera.

Repetir

Escribir "Este bucle nunca terminará"

Hasta Que falso

### Resumen

Un bucle Repetir ejecuta un bloque de código al menos una vez y repite la ejecución hasta que una condición específica se vuelve verdadera.

## Estructura algorítmica repetitiva Para(for)

Un bucle Para es una estructura de control que permite repetir un bloque de código un número determinado de veces. Se utiliza cuando se conoce de antemano cuántas veces se necesita repetir la operación.

**Importancia**: Los bucles Para son esenciales para iteraciones controladas, como recorrer rangos de números, acceder a elementos de una lista por índice, o repetir una operación un número específico de veces.

**Ejemplo del mundo real**: Caminar 10 pasos, contar de 1 a 100, o repetir una tarea específica un número conocido de veces.

### Ejercicio

* **`Para`:** Palabra clave que indica el inicio del bucle.
* **`variable`:** Variable de control del bucle que toma valores desde `valor\_inicial` hasta `valor\_final`.
* **`valor\_inicial`:** Valor inicial de la variable de control.
* **`valor\_final`:** Valor final que alcanzará la variable de control.
* **`Con Paso paso`:** (Opcional) Indica el incremento o decremento de la variable de control en cada iteración. Por defecto es 1.
* **`Hacer`:** Palabra clave que indica el inicio del bloque de código que se repetirá.
* **`FinPara`:** Palabra clave que indica el fin del bucle.

**Explicación:**

* La **variable de control** se inicializa con valor\_inicial.
* En cada iteración, la variable de control se incrementa (o decrementa) por paso.
* Si la variable de control es menor o igual a valor\_final (para pasos positivos) o mayor o igual a valor\_final (para pasos negativos), el bucle se ejecuta.
* El bucle termina cuando la variable de control excede valor\_final.

**Imprimir números del 1 al 10**:

**Para i <- 1 Hasta 10 Hacer**

**Escribir i**

**FinPara**

**Solución:** i toma valores desde 1 hasta 10. En cada iteración, i se incrementa en 1 y se imprime su valor.

### Errores comunes

**Valores Incorrectos de Inicio y Fin**: Asegúrate de que los valores inicial y final son correctos y el incremento es adecuado.

**Para i <- 10 Hasta 1 Hacer**

**Escribir i**

**FinPara**

**Paso Incorrecto**: Verifica que el paso sea coherente con el rango.

**Para i <- 1 Hasta 10 Con Paso 0 Hacer**

**Escribir i**

**FinPara**

### Resumen

Un bucle Para se utiliza cuando se conoce de antemano cuántas veces se necesita repetir una operación.

La variable de control toma valores desde el inicial hasta el final, con un paso específico.