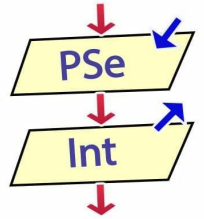
CODO A CODO INICIAL
Clase 9

Algoritmia 9

BA

Estructuras repetitivas I



Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 08

Algoritmia 8 - Estructuras condicionales IV

- Operadores lógicos: AND (y), OR (o), NOT (no).
- Tablas de verdad.
- Uso de condicionales junto a operadores lógicos.
- Precedencia de operadores.

Clase 09

Algoritmia 9 - Estructuras Repetitivas I

- Estructuras repetitivas.
- Uso de "Repetir...Hasta Que"
- Condicionales en las estructuras repetitivas.

Clase 10

Algoritmia 10 - Estructuras Repetitivas II

- Contador.
- Acumulador.
- Bandera.
- Combinación de bucles y condicionales. Ejemplos.

Estructuras repetitivas

Las **estructuras repetitivas** son elementos clave que permiten la ejecución repetida de un bloque de código mientras se cumpla una **condición** específica.

Estas estructuras son esenciales para la implementación de bucles en algoritmos, ofreciendo una forma eficiente de realizar tareas repetitivas y controlar dinámicamente la ejecución del programa en **PSeInt**.

Existen distintas estructuras repetitivas en PSeint, aquí trabajaremos con **Mientras...Hacer** .

Estructuras repetitivas | Conceptos importantes

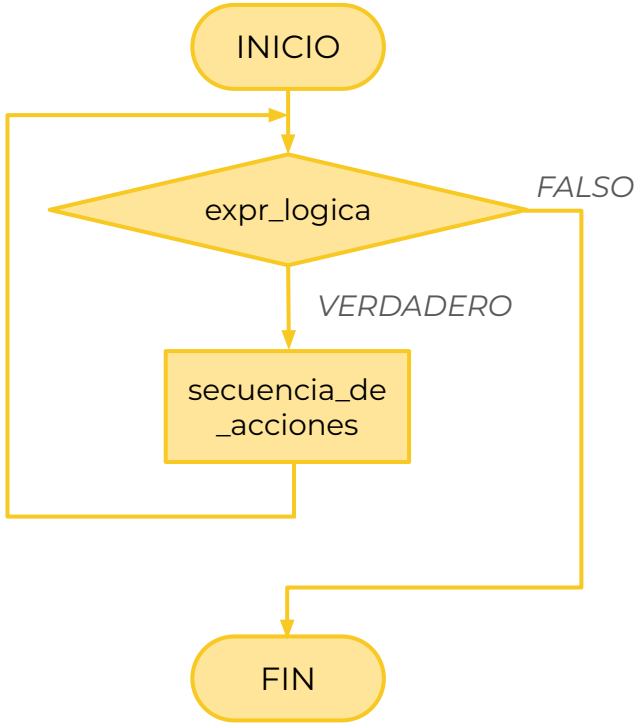
- **Bucle:** Es una estructura de control de flujo que repite un bloque de instrucciones un número determinado de veces. Es un sinónimo de *estructura repetitiva*.
- **Iteración:** Llamamos iteración a cada una de las veces que se produce la ejecución del bloque de instrucciones que contiene el bucle.
- **Condición:** Su valor de verdad determina si el bucle se repite una vez mas, o si ha finalizado. Todo bucle tiene que llevar asociada una condición, y debemos asegurarnos de que dicha condición en algún momento sea falsa, para finalizar la ejecución del bucle.

Estructura Mientras...Hacer

En PSeint, la estructura repetitiva **Mientras... Hacer** se utiliza para crear un bucle que ejecuta un conjunto de instrucciones mientras una condición específica sea verdadera. La sintaxis básica de esta estructura es la siguiente:

```
Mientras expresion_logica Hacer
    secuencia_de_acciones
Fin Mientras
```

expresion_logica es evaluada en cada iteración. En caso de ser verdadera se ejecutará la *secuencia_de_acciones*.



Estructura Mientras...Hacer | Caso práctico

Debemos realizar un programa que escriba en pantalla los números de 1 a 10. Conocemos la instrucción **Escribir**, que repetida 10 veces cumple con la tarea.

Durante el proceso de creación de programas, es frecuente encontrar que una o más operaciones, a veces con cambios mínimos, deben repetirse muchas veces.

Si copiamos y pegamos las instrucciones que se repiten nos quedaría algo similar a la imagen de la derecha:

```
Algoritmo contarHasta10

    Escribir 1
    Escribir 2
    Escribir 3
    Escribir 4
    Escribir 5
    Escribir 6
    Escribir 7
    Escribir 8
    Escribir 9
    Escribir 10

FinAlgoritmo
```

Estructura Mientras...Hacer | Caso práctico

Si tuviésemos que realizar la misma tarea con 100 o 1000 números este enfoque deja de ser práctico.

Aquí es donde aparece la estructura **Mientras...Hacer**:

```
Algoritmo contarHasta100

    num = 1
    Mientras num <= 10 Hacer
        Escribir num
        num = num + 1
    Fin Mientras

FinAlgoritmo
```

A

B

C

Esta estructura requiere de una **variable** para **contar** la cantidad de iteraciones **(A)**, en este caso se le asigna el valor inicial **1**.

La estructura **Mientras** num <= 10 **Hacer** **(B)** contiene la condición que controla el bucle.

Mientras **num** sea menor o igual a 10 se repiten el bloque de instrucciones **(C)**. La primera muestra el valor de la variable y la segunda **incrementa** el valor de la variable en 1.

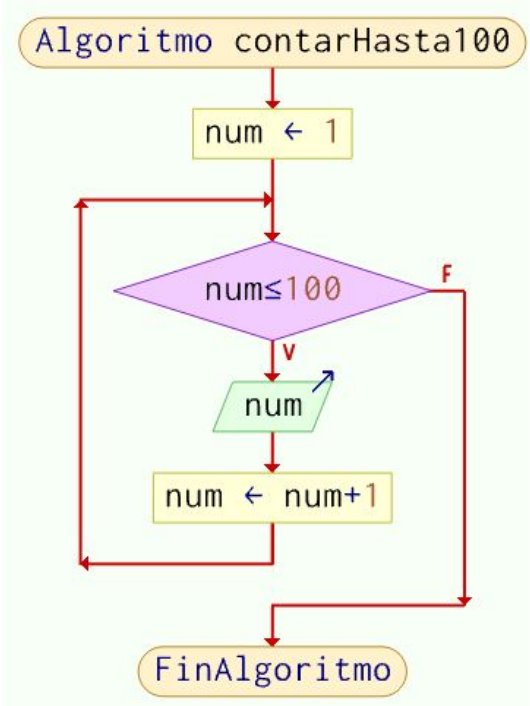
Es **importante incrementar la variable** para evitar un *ciclo infinito*.

Estructura Mientras...Hacer | Caso práctico

El diagrama de flujo comienza asignando el valor 1 a la variable ***num***.

El **rombo** representa la condición, con dos salidas:

- La salida por *falso* se produce cuando **no se cumpla** la condición num <= 100, y allí terminará el algoritmo.
- La salida por *verdadero* tiene lugar **mientras que se cumple** la condición, es decir para valores entre 1 y 100. Se repetirán las instrucciones para mostrar el número e incrementarlo cada vez que la condición se cumpla.



Estructura Mientras...Hacer | Consideraciones

Algunas consideraciones a partir del ejemplo anterior:

- Las instrucciones se realizan siempre y cuando la condición evaluada (expresión lógica) sea **verdadera**.
- Cuando la expresión lógica es verdadera se ejecuta el **cuerpo del bucle** y luego se vuelve a evaluar la expresión lógica. Este proceso se repite una y otra vez mientras la expresión lógica (condición evaluada) sea **verdadera**, para salir del bucle la condición debe ser **falsa**.
- Si la condición se evalúa **falsa**, no se toma ninguna acción y el programa prosigue con la siguiente instrucción.

Estructura Mientras...Hacer | Consideraciones

- No siempre se conoce de antemano el número de iteraciones.
- La condición se verifica **antes** de la ejecución del cuerpo del bucle.
- Las instrucciones del cuerpo del bucle se ejecutan en forma repetitiva si la condición es verdadera.
- Si la expresión a evaluar es falsa, **nunca ingresamos al bucle** y las instrucciones de la estructura no se ejecutan.
- Para finalizar el bucle se debe modificar el valor de la condición, haciendo que sea falso, sino corremos el riesgo de caer en un bucle infinito.

Estructura Mientras...Hacer | Validaciones

Esta estructura se puede utilizar para **validar** el ingreso de un dato, es decir, permitir solamente datos que para el programa se consideren válidos.

En el siguiente ejemplo se permite el ingreso **únicamente de edades mayores o iguales a 18 años**.

Deben descartarse todos aquellos números menores a 18. **Mientras el número ingresado** sea menor a 18, se produce una nueva iteración y volvemos a pedirlo:

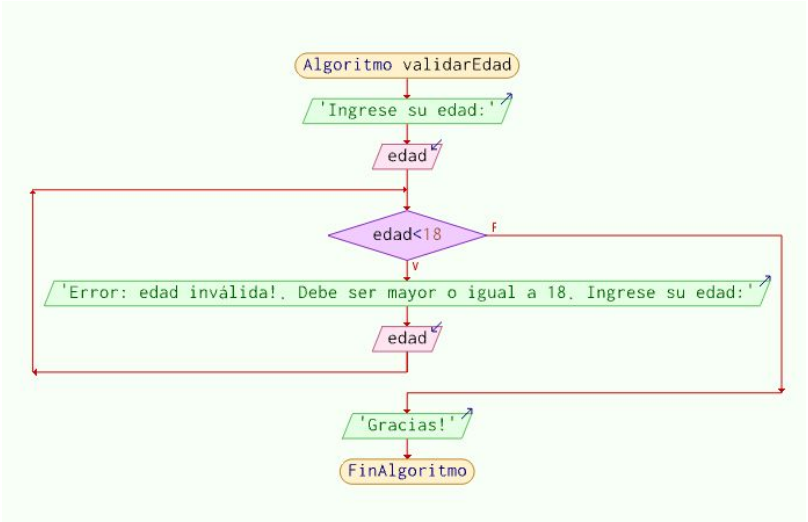
```
Algoritmo validarEdad
     Escribir "Ingrese su edad:"
     Leer edad
     Mientras edad < 18 Hacer
          Escribir "Error: edad
inválida!. Debe ser mayor o igual a 18.
Ingrese su edad:"
          Leer edad
     FinMientras
     Escribir "Gracias!"
FinAlgoritmo
```

Estructura Mientras...Hacer | Validaciones

Al ejecutar el ejemplo anterior vemos que no podemos escribir edades menores a 18 años:

```
*** Ejecución Iniciada. ***
Ingrese su edad:
> 15
Error: edad inválida!. Debe ser mayor o igual a 18. Ingrese su edad:
> 8
Error: edad inválida!. Debe ser mayor o igual a 18. Ingrese su edad:
> 25
Gracias!
*** Ejecución Finalizada. ***
```

El programa continuará pidiendo la edad **mientras** no ingresemos una edad válida (edad mayor o igual a 18).



Estructura Mientras...Hacer | Validaciones

Otra práctica común es validar el ingreso de un **rango de valores**, restringiendo el ingreso a valores que no cumplan ciertos requisitos.

En ejemplo se permite **solamente que ingresen notas entre 1 y 10**.

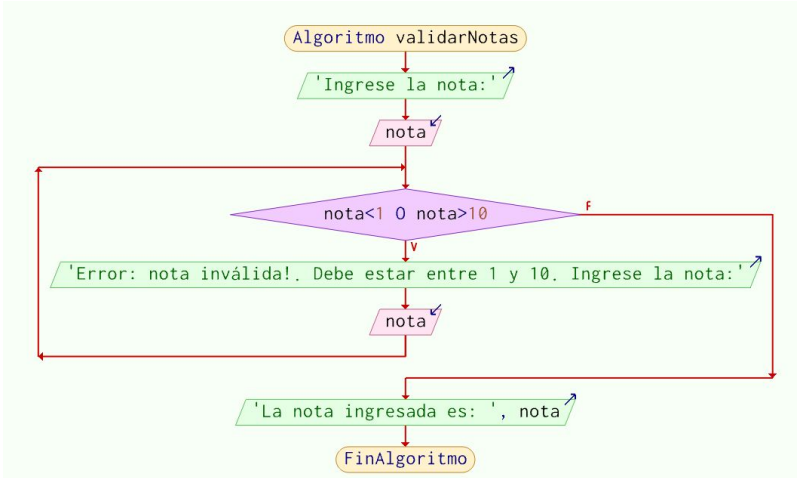
Deben descartarse todos aquellos números menores a 1 pero a la vez aquellos que sean mayores a 10, es decir que **mientras el número ingresado** sea menor a 1 **o** mayor a 10 debemos volver a pedirlo:

```
Algoritmo validarNotas
     Escribir "Ingrese la nota:"
     Leer nota
     Mientras nota < 1 o nota > 10 Hacer
          Escribir "Error: nota inválida!.
Debe estar entre 1 y 10. Ingrese la nota:"
          Leer nota
     FinMientras
     Escribir "La nota ingresada es: ",
nota
FinAlgoritmo
```

Estructura Mientras...Hacer | Validaciones

Ejecutando el código, comprobamos que no podremos escribir notas fuera del rango 1-10:

```
*** Ejecución Iniciada. ***
Ingrese la nota:
> 11
Error: nota inválida!. Debe estar entre 1 y 10. Ingrese la nota:
> 0
Error: nota inválida!. Debe estar entre 1 y 10. Ingrese la nota:
> 7
La nota ingresada es: 7
*** Ejecución Finalizada. ***
```



El programa continúa pidiendo notas **mientras** no se ingrese una nota válida (entre 1 y 10 inclusive).

Desafíos

Desafío 1: Ensalada de frutas

Escribir un programa que pida al usuario nombre de frutas. Aceptar únicamente los valores “manzana”, “pera”, “pomelo” o “naranja”.

Si se ingresa otro valor, indicar el error y solicitar un nuevo valor.

Si se ingresa la palabra “fin” terminar el programa.

Desafío 2: El valor más pequeño

Escribir un programa que pida al usuario números positivos hasta que el valor ingresado sea igual a cero.

Mostrar en la pantalla el valor más pequeño de todos los que se ingresaron (sin considerar el cero que se ingresó para finalizar el bucle).

Material extra

Artículos de interés

Material extra:

- Condiciones repetitivas (1: Mientras) | AprendeAProgramar.com
- Estructura Repetitiva MIENTRAS (While) con PSeInt | Pedro Antonio Villalta
- Ejercicios con estructura repetitiva Mientras – Algoritmos en Pseint | Algoritmos y Algo Más

Videos:

- Ciclo mientras pseint ejemplos | AlgoritmoDeTarea
- Estructura Repetitiva MIENTRAS (WHILE) | Introducción a los ALGORITMOS y la PROGRAMACIÓN | TodoCode
- PSeInt. FUNCIÓN MIENTRAS. Sumatoria. Explicación detallada | TecnoMáticas

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.
- Realizar el Ejercicio de Repaso.

Todo en el Aula Virtual.

Muchas gracias por tu atención.

Nos vemos pronto.