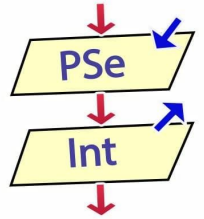
CODO A CODO INICIAL
Clase 10

Algoritmia 10

BA

Estructuras repetitivas II



Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 09

Algoritmia 9 - Estructuras Repetitivas I

- Definición y explicación de las estructuras repetitivas en programación.
- Uso de "Mientras...Hacer"

Clase 10

Algoritmia 10 - Estructuras Repetitivas II

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

Clase 11/12

Algoritmia 11/12 - Síntesis y repaso

- Integración de temas.
- Tipos de datos y variables.
- Entrada y salida de datos.
- Estructuras condicionales.
- Operadores lógicos
- Estructuras repetitivas.

Contador

En PSeInt, un **contador** dentro de una estructura repetitiva **es una variable que se incrementa o decrementa en cada iteración del bucle.**

Se utiliza para contar el número de repeticiones y controlar la ejecución del bucle. Se inicializa antes de la estructura repetitiva y se actualiza dentro del bucle. Su valor nuevo es el resultado del valor anterior más una constante (en general el valor 1).
Los contadores son valiosos para controlar el flujo de ejecución y realizar tareas repetitivas una cantidad específica de veces, evitando caer en un ciclo infinito.

Contador | Ejemplo de uso

```
Algoritmo ejemploContador

    cont = 1
    tabla = 5
    Escribir "Tabla del: ", tabla
    Mientras cont <= 10 Hacer
        Escribir cont, " x ", tabla, "
= ", cont * tabla
        cont = cont + 1
    Fin Mientras

FinAlgoritmo
```

A

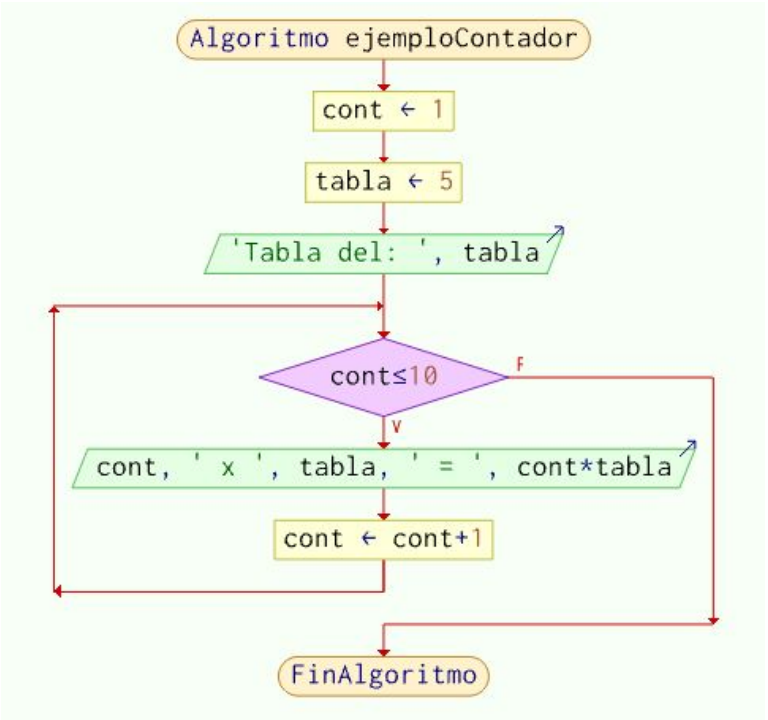
B

C

El objetivo de este programa es mostrar la tabla del 5, desde el 1 al 10.
La variable **cont** actúa como un *contador de iteraciones* del bucle **Mientras**.
Se inicializa en 1 **(A)** y en cada iteración se incrementa de uno en uno **(B)** hasta llegar a 10 **(C)**, momento en el cual se sale del bucle.
Si se imprimiera la variable **cont** luego de la ejeciòn del bucle **valdrìa 11**, ya que ese es un valor que no cumple con la condición.

Contador | Ejemplo de uso

```
*** Ejecución Iniciada. ***
Tabla del: 5
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
4 x 5 = 20
5 x 5 = 25
6 x 5 = 30
7 x 5 = 35
8 x 5 = 40
9 x 5 = 45
10 x 5 = 50
*** Ejecución Finalizada. ***
```



Acumulador

Un **acumulador** dentro de una estructura repetitiva **es una variable que se utiliza para sumar o acumular valores en cada iteración del bucle**. Este tipo de variable se inicializa antes de entrar al bucle y se actualiza en cada repetición. Su valor nuevo es el resultado del valor anterior más una variable.

Los acumuladores son útiles para llevar un seguimiento de la suma total de valores y son comúnmente utilizados para calcular promedios, totales o realizar otras operaciones que involucren la acumulación de datos durante la ejecución del programa.

Acumulador | Ejemplo de uso

```
Algoritmo ejemploAcumulador

    total = 0
    precio = 0
    cont = 1

    Mientras cont <= 5 Hacer
        Escribir "Ingrese un precio: "
        Leer precio
        total = total + precio
        cont = cont + 1
    FinMientras
    Escribir "El total es: ", total

FinAlgoritmo
```

A

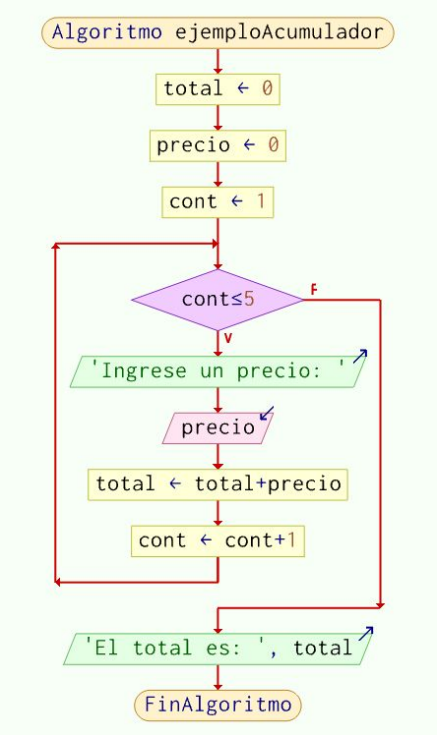
B

C

El objetivo de este programa es sumar los 5 valores que se ingresan por teclado y mostrar el resultado.
La variable **total** actúa como un *acumulador* e irá almacenando los resultados parciales de cada iteración del bucle **Mientras**.
Se inicializa en 0 **(A)** y en cada iteración se suma su valor con **precio** **(B)** hasta llegar a 5, momento en el cual se sale del bucle.
Finalmente se imprime el total acumulado **(C)**.

Acumulador | Ejemplo de uso

```
*** Ejecución Iniciada. ***
Ingrese un precio:
> 150
Ingrese un precio:
> 200
Ingrese un precio:
> 50
Ingrese un precio:
> 350
Ingrese un precio:
> 45
El total es: 795
*** Ejecución Finalizada. ***
```



Contador y Acumulador | Ejemplo de uso

Escribir un programa que calcule el promedio de las notas ingresadas.
Se sale del programa cuando el usuario ingresa 0 como nota.
Imprimir la suma, cantidad y promedio de las notas.

Debemos tener en cuenta:

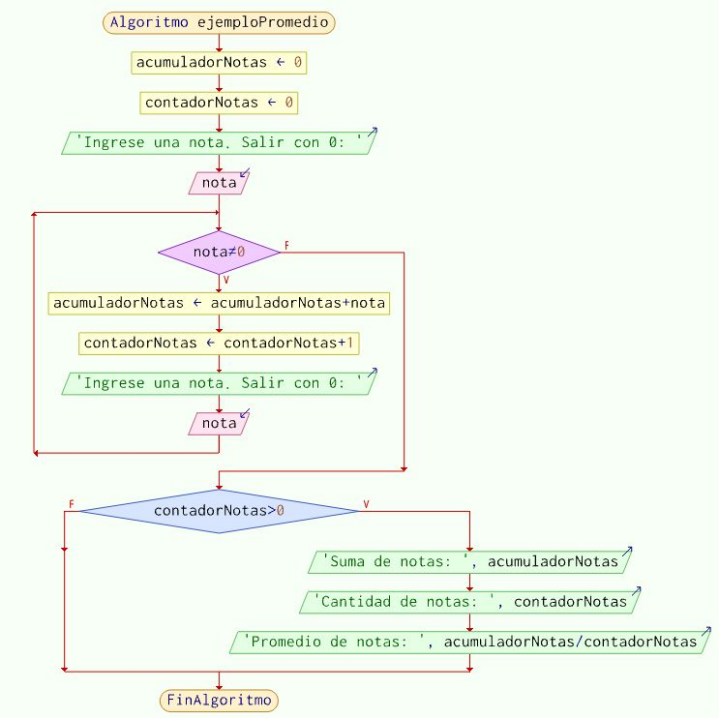
- No sabemos de antemano la cantidad de notas que se van a ingresar, pero sì sabemos que se sale con 0.
- Se deben acumular y contar las notas.
- Verificar que el cálculo del promedio sea realizado fuera del bucle
- Evitar la divisiòn por cero

Contador y Acumulador | Ejemplo de uso

```
Algoritmo ejemploPromedio

	acumuladorNotas = 0
	contadorNotas = 0

	Escribir "Ingrese una nota. Salir con 0: "
	Leer nota

	Mientras nota <> 0 Hacer
		acumuladorNotas = acumuladorNotas + nota
		contadorNotas = contadorNotas  + 1
		Escribir "Ingrese una nota. Salir con 0: "
		Leer nota
	FinMientras

	Si contadorNotas > 0 Entonces
		Escribir "Suma de notas: ", acumuladorNotas
		Escribir "Cantidad de notas: ", contadorNotas
		Escribir "Promedio de notas: ", acumuladorNotas/contadorNotas
	FinSi
FinAlgoritmo
```

Contador y Acumulador | Ejemplo de uso

```
*** Ejecución Iniciada. ***
Ingrese una nota. Salir con 0:
> 4
Ingrese una nota. Salir con 0:
> 10
Ingrese una nota. Salir con 0:
> 7
Ingrese una nota. Salir con 0:
> 0
Suma de notas: 21
Cantidad de notas: 3
Promedio de notas: 7
*** Ejecución Finalizada. ***
```



Bandera

Una variable **bandera** es una variable *booleana* (es decir, que puede tener dos valores: verdadero o falso) que se utiliza para indicar el estado de una condición o evento en un programa. Su nombre, "bandera" o "flag", sugiere que se utiliza como una señal para marcar o indicar algo.
La variable bandera generalmente comienza con un valor inicial, y su valor se modifica a medida que se cumple o deja de cumplirse cierta condición.
Su propósito principal es actuar como una señal o indicador que puede cambiar su estado durante la ejecución del programa.
Nos permite controlar ciclos donde no esta definido el numero de iteraciones.

Bandera | Ejemplo de uso

Desarrolle un programa en PSeInt que simule un sistema de acceso con clave. El programa debe permitir al usuario ingresar una clave y verificar si es correcta. Se establece que la clave correcta es "secreto". El usuario tiene hasta tres intentos para ingresar la clave correcta.

Debemos tener en cuenta:

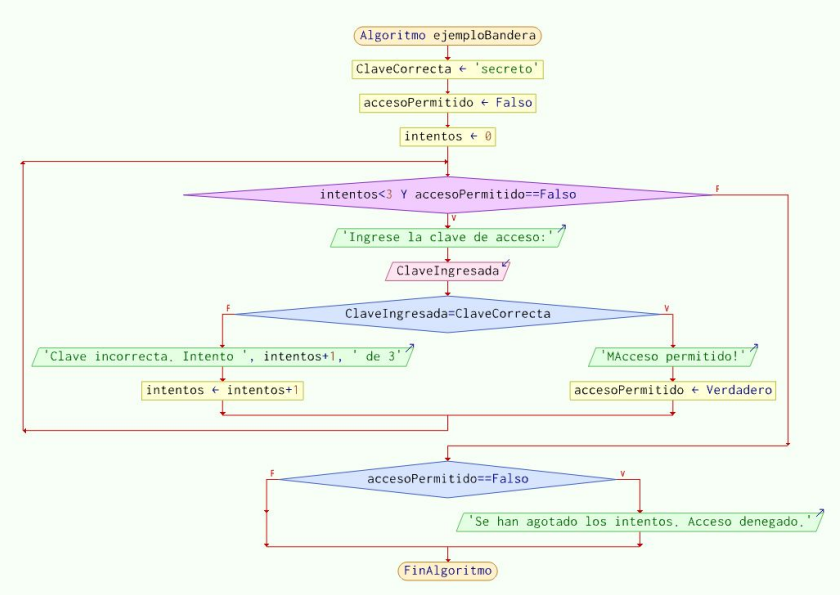
- Utilizar una variable bandera para controlar si el usuario ha ingresado la clave correcta.
- Utilizar una variable para contabilizar el número de intentos realizados.
- Desarrollar mensajes informativos adecuados, mostrando además la cantidad de intentos.
- Si el usuario ingresa la clave correcta, se cambiará el estado de la bandera a Verdadero y se imprimirá un mensaje de "¡Acceso permitido!". Si la clave es incorrecta, se informará al usuario y se incrementará el contador de intentos.

Bandera | Ejemplo de uso

```
Algoritmo ejemploBandera

	claveCorrecta = "secreto"
	accesoPermitido = Falso
	intentos = 0

	Mientras intentos < 3 Y AccesoPermitido == Falso Hacer
		Escribir "Ingrese la clave de acceso:"
		Leer claveIngresada

		Si ClaveIngresada = ClaveCorrecta Entonces
			Escribir "¡Acceso permitido!"
			accesoPermitido = Verdadero
		Sino
			Escribir "Clave incorrecta. Intento ", intentos + 1, " de 3"
			intentos = intentos + 1
		Fin Si
	Fin Mientras

	Si accesoPermitido == Falso Entonces
		Escribir "Se han agotado los intentos. Acceso denegado."
	Fin Si

FinAlgoritmo
```

Bandera | Ejemplo de uso

```
*** Ejecución Iniciada. ***
Ingrese la clave de acceso:
> Hola
Clave incorrecta. Intento 1 de 3
Ingrese la clave de acceso:
> 44
Clave incorrecta. Intento 2 de 3
Ingrese la clave de acceso:
> secreto
¡Acceso permitido!
*** Ejecución Finalizada. ***
```



Contadores y acumuladores | Más ejemplos

Escribir un programa que lea números enteros hasta teclear 0 (cero). Al finalizar el programa se debe mostrar el máximo número ingresado, el mínimo, y el promedio de todos ellos.

Debemos tener en cuenta:

- El bucle se ejecutará mientras que el usuario no ingrese 0
- El número debe pedirse antes de entrar al bucle y dentro del bucle (para evitar caer en un ciclo infinito).
- Inicializar adecuadamente las variables para obtener el mínimo, el máximo y el promedio (considerar cómo se obtiene y dónde calcularlo)

Contadores y acumuladores | Más ejemplos

```
Algoritmo calculoNumeros

    suma = 0
A   cuenta = 0
    maximo = 0

B   Escribir "Ingrese un número, salir con 0:"
    Leer num

C   Mientras num <> 0 Hacer
        suma = suma + num // Acumulador
        Si num > maximo Entonces
D           maximo = num
        Fin Si
        cuenta = cuenta + 1
```

Este código en PSeInt solicita al usuario ingresar números hasta que ingrese el número 0.
Las variables suma, cuenta y maximo son inicializadas en 0 para guardar los valores necesario para los cálculos pedidos **(A)**.
Luego se pide un número al usuario **(B)** y **mientras que el usuario no haya ingresado un 0 (C)** se acumula el valor en la variable **suma** y se determina el número máximo con un condicional **(D)**.

Contadores y acumuladores | Más ejemplos

```
        cuenta = cuenta + 1
    Escribir "Ingrese un número, salir con 0:"
        Leer num
    FinMientras
    Si cuenta > 0 Entonces
        Escribir "El promedio es: ", suma/cuenta
        Escribir "El número máximo es: ", maximo
    SiNo
        Escribir "No se han ingresado números"
    Fin Si

FinAlgoritmo
```

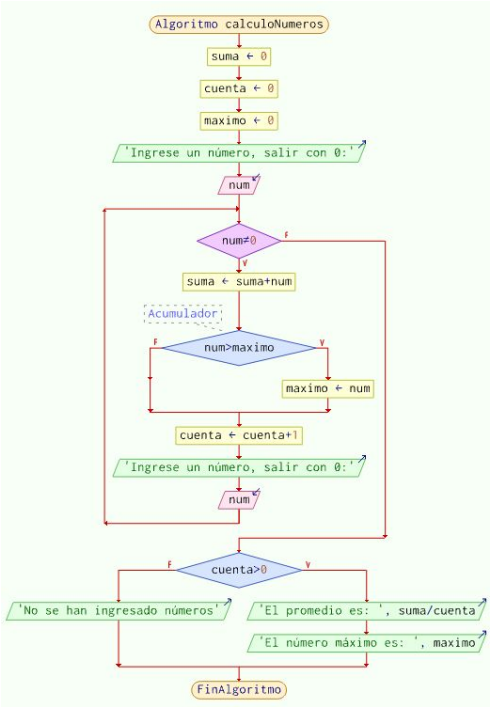
Se incrementa la variable **cuenta** en 1 para llevar la cuenta de los números ingresados **(E)**.
Luego se vuelve a pedir un número al usuario **(F)** y cierra el bucle. Ese valor será el que modifique la condición de salida del bucle.
Al salir del bucle se realiza una verificación **(G)** para evitar la división por 0 si el usuario ingresó 0 por primera vez:

- Si **cuenta** es mayor que 0, se muestra el promedio y el número máximo ingresado
- Si **cuenta** es igual a 0, se informa al usuario que no se han ingresado números.

Contadores y acumuladores | Más ejemplos

```
*** Ejecución Iniciada. ***
Ingrese un número, salir con 0:
> 2
Ingrese un número, salir con 0:
> 4
Ingrese un número, salir con 0:
> 6
Ingrese un número, salir con 0:
> 0
El promedio es: 4
El número máximo es: 6
*** Ejecución Finalizada. ***
```

```
*** Ejecución Iniciada. ***
Ingrese un número, salir con 0:
> 0
No se han ingresado números
*** Ejecución Finalizada. ***
```



A través de estas salidas por pantalla vemos còmo se comporta el programa cuando ingresamos datos y cuando ingresamos 0 por primera vez-

Contadores y acumuladores | Más ejemplos

Escribir un programa que calcule el cuadrado de los 9 primeros números naturales e imprima por pantalla el número seguido de su cuadrado. Ejemplo: “2 elevado al cuadrado es 4”, y así sucesivamente.

Debemos tener en cuenta:

- El bucle se ejecutará 9 veces (los primeros 9 números naturales)
- No se piden datos al usuario y se debe calcular el cuadrado del número en cada iteración (número multiplicado por sí mismo)
- En cada iteración se debe mostrar el número y su cuadrado
- Inicializar adecuadamente las variables para obtener el número y su cuadrado

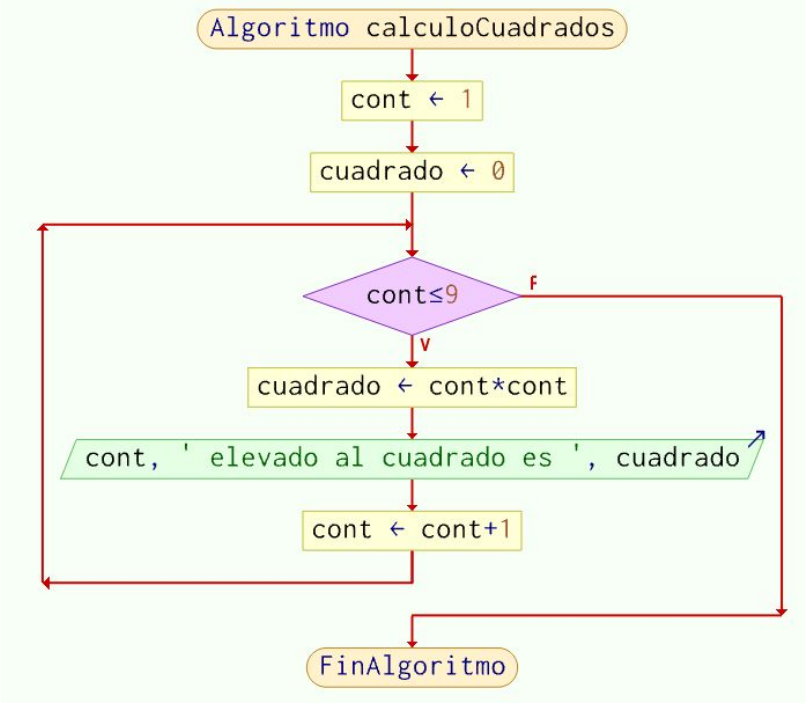
Contadores y acumuladores | Más ejemplos

```
Algoritmo calculoCuadrados

    cont = 1
A   cuadrado = 0
B   Mientras cont <= 9 Hacer
        cuadrado = cont * cont
C       Escribir cont, " elevado al
cuadrado es ", cuadrado
        cont = cont + 1
    Fin Mientras

FinAlgoritmo
```

Este programa permite calcular y mostrar los cuadrados de los números del 1 al 9 en la pantalla.
Se inicializan las variables **cont** (contador de iteraciones) y **cuadrado** (almacenará el resultado) **(A)**..
El bucle Mientras se ejecutará mientras el valor de **cont** sea menor o igual a 9 **(B)**. y en su interior calcula el cuadrado del número actual (cont * cont) y se almacena en la variable cuadrado. Luego, se muestra en la pantalla el número actual (**cont**), la expresión "elevado al cuadrado es", y el valor del cuadrado (**cuadrado**).
Finalmente se incrementa el valor de cont en 1 para pasar al siguiente número en la siguiente iteración del bucle **(C)**., que continúa ejecutándose hasta que **cont** sea mayor que 9, momento en el cual el bucle se detiene.

Contadores y acumuladores | Más ejemplos

```
*** Ejecución Iniciada. ***
1 elevado al cuadrado es 1
2 elevado al cuadrado es 4
3 elevado al cuadrado es 9
4 elevado al cuadrado es 16
5 elevado al cuadrado es 25
6 elevado al cuadrado es 36
7 elevado al cuadrado es 49
8 elevado al cuadrado es 64
9 elevado al cuadrado es 81
*** Ejecución Finalizada. ***
```



Desafíos

Desafío 1: ¿Piedra, papel o tijera?

- Con todo lo aprendido hasta ahora podemos pensar en un algoritmo que nos permita jugar a “piedra, papel o tijera”. El juego corre hasta que el usuario decida que no quiere continuar jugando (ingresando “s” o “n”). Además, debe contar cuantas partidas gano el usuario y cuantas la computadora.
- Como desafìo adicional podemos pensar qué modificaciones debemos hacer si deseamos que se realice una serie de 6 partidas fijas. ¿Y si pudiéramos decidir que el número de partidas se ingrese previamente por el usuario?

Desafío 2: Un menú dinámico

- Realicemos un menú dinámico, donde tengamos diferentes opciones:
  - Mostrar los números pares hasta un número dado.
  - Mostrar la tabla de multiplicar para un número dado.
  - Mostrar un número elevado a una potencia
  - Una opción para Salir

Material extra

Artículos de interés

Material extra:

- Algoritmos: Variables, contadores y acumuladores | Kalim Al Razif
- Manual de Programacion Logica | Lic. Ricardo Saucedo (pp 29-33)
-

Videos:

- Contadores y Acumuladores en PseInt | Elfar Didier Morantes Sanchez
- Banderas o Flags Diagrama de flujo | Marcos Rios

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