Apellido y Nombre: …… *Fecha: ……/……/……*

CONCEPTOS TEÓRICOS A TENER EN CUENTA

Las computadoras están especialmente diseñadas para todas aquellas aplicaciones en las que una operación o conjunto de ellas deben repetirse muchas veces.

La cantidad de iteraciones o repeticiones puede ser fija (previamente determinada por el programador) o variable (depende de algún dato o evento del programa).

Las sentencias que permiten especificar la repetición de un conjunto de instrucciones son:

- ***Para*** (PARA-FIN_PARA)
- ***Mientras*** (MIENTRAS-FIN_MIENTRAS)
- ***Repetir*** (REPETIR-HASTA_QUE)

PROBLEMAS REPETITIVOS O CÍCLICOS: *son aquellos que para resolverse aplican un conjunto de acciones repetidamente (un número definido o indefinido de veces).*

BUCLES: *estructuras de control que permiten repetir una secuencia de instrucciones.*

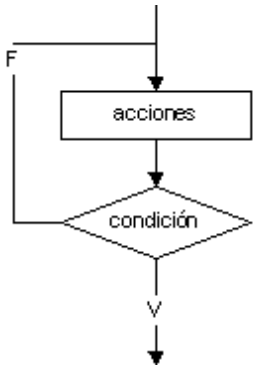
ITERACIÓN: *la iteración o repetición se refiere a la ejecución de un conjunto de acciones.*

ESTRUCTURAS DE CONTROL REPETITIVAS

DIAGRAMA DE FLUJO Y SINTAXIS EN PSEUDOCÓDIGO: REPETIR, MIENTRAS Y PARA.

REPETIR

```
REPETIR
    acciones
HASTA_QUE condición
```

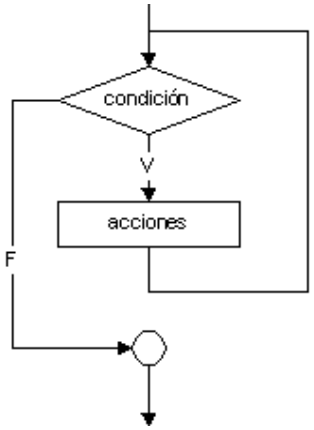


Funcionamiento:

1. Se ejecuta el bloque de ACCIONES incluido en el bucle.
2. Se evalúa la CONDICIÓN.
3. Si la CONDICIÓN toma el valor Falso, se retorna al paso 1, si toma valor Verdadero, se sigue al paso 4.
4. Sale del bucle.

MIENTRAS

```
MIENTRAS condición HACER
    acciones
FIN_MIENTRAS
```

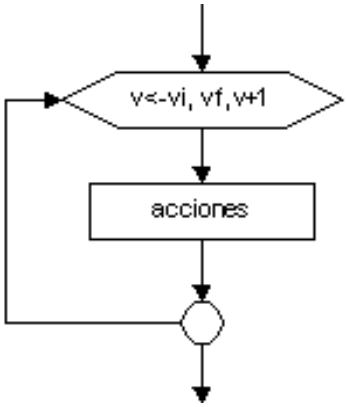


Funcionamiento:

1. Se evalúa la CONDICIÓN.
2. Si la CONDICIÓN toma el valor Verdadero, se sigue con el paso 3, si la CONDICIÓN toma valor FALSO, se sigue al paso 4.
3. Se ejecuta el bloque de ACCIONES incluido en el bucle y se retorna al paso 1.
4. Sale del bucle.

PARA

```
PARA v DESDE vi HASTA vf CON PASO n HACER
    acciones
FIN_PARA
```



Funcionamiento:

1. Se asigna a la variable de control o índice el *valor inicial* (***vi***)
2. Si la CONDICIÓN (***vi*** <= ***vf***) toma el valor Verdadero, se sigue con el paso 3, si la CONDICIÓN (***vi*** <= ***vf***) toma valor FALSO, se sigue al paso 5.
3. Por defecto, se incrementa (***v+1***) o decrementa (***v-1***) el índice en una unidad, aunque puede especificarse otro valor.
4. Se ejecuta el bloque de ACCIONES incluido en el bucle, se retorna al paso 2.
5. Sale del bucle.

EJERCICIOS RESUELTOS

1. Diseñe un algoritmo (diagrama de flujo y pseudocódigo) que sume 30 valores ingresados por el usuario.

¿Cuál es el problema que debo resolver? — *Calcular la suma de 30 valores*

¿Qué datos necesito para resolver el problema? — *30 valores generados por el usuario*

¿Cuál es el resultado que debo obtener? — *La suma de 30 valores*

¿Cuáles son los pasos que debo realizar para obtener la solución? —
1) Obtener los 30 valores
2) Sumar cada uno de los 30 valores ingresados
3) Mostrar el resultado calculado

¿Cómo realizo 30 lecturas y 30 sumas? Tendré que escribir 60 instrucciones?

Ya sé, escribo una lectura y una suma, con un bucle se repetirán 30 veces

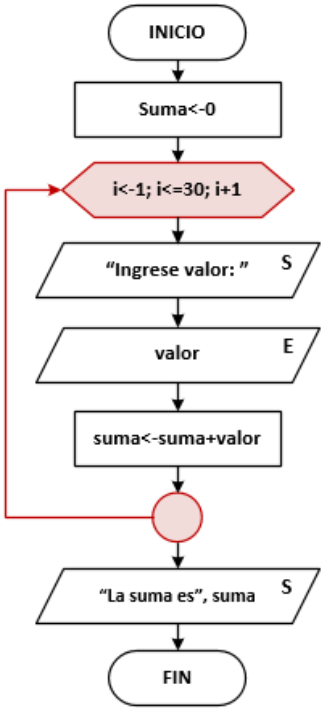
Pero, qué tipo de bucle usaré?

Mmmmm Repetir, Mientras o Para?

Tengo 30 valores de entrada, y debo calcular 30 sumas

Ya sé, cuando la cantidad de veces que debo aplicar una operación es fija, uso PARA!!!!

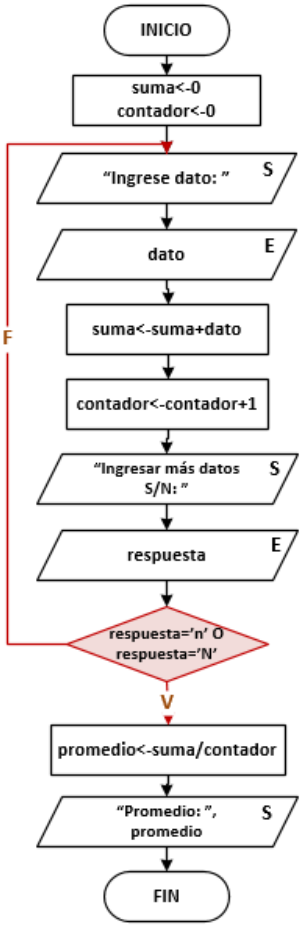
```
PROGRAMA Ejemplo_1
VARIABLES
    suma, i, valor: ENTERO
INICIO
    suma ← 0
    PARA i DESDE 1 HASTA 30 HACER
        ESCRIBIR "Ingrese valor:"
        LEER valor
        suma ← suma + valor
    FIN_PARA
    ESCRIBIR "la suma es:" suma
FIN
```



ACUMULADOR: *es una variable cuya misión es almacenar cantidades variables resultantes de sumas o productos sucesivos.*

El algoritmo tiene por objetivo **sumar 30 valores** ingresados por el usuario, lo que implica realizar 30 operaciones de lectura y 30 operaciones de suma. Este conjunto de operaciones puede resumirse aplicando estructuras repetitivas, en este caso, utilizando la estructura **PARA** ya que se conoce la **cantidad de repeticiones** a realizar. En cada iteración (repetición), la variable *valor* se carga con el dato ingresado por el usuario e inmediatamente se adiciona a la variable *suma*. La variable suma, cuyo valor inicial es cero (neutro de la operación de adición), funciona como un **acumulador** de los valores.

2. Diseñe un algoritmo (diagrama de flujo y pseudocódigo) que calcule el promedio de los valores ingresados por el usuario. Considere que el ingreso finaliza a pedido del usuario.




```
PROGRAMA Ejemplo_2
VARIABLES
    suma, contador: ENTERO
    promedio, dato: REAL
    respuesta: CARACTER
INICIO
    suma←0
    contador←0
    REPETIR
        ESCRIBIR "Ingrese dato:"
        LEER dato
        suma←suma + dato
        contador←contador + 1
        ESCRIBIR "Ingresar más datos s/n:"
        LEER respuesta
    HASTA_QUE respuesta='n' O respuesta='N'
    promedio ← suma/contador
    ESCRIBIR "Promedio:" promedio
FIN
```

CONTADOR: *es una variable cuyo valor se incrementa o decrementa en una cantidad constante en cada iteración. El contador puede ser positivo o negativo.*

CENTINELA: *es aquel valor especial que, al generarse en el bucle, permite finalizar las iteraciones.*

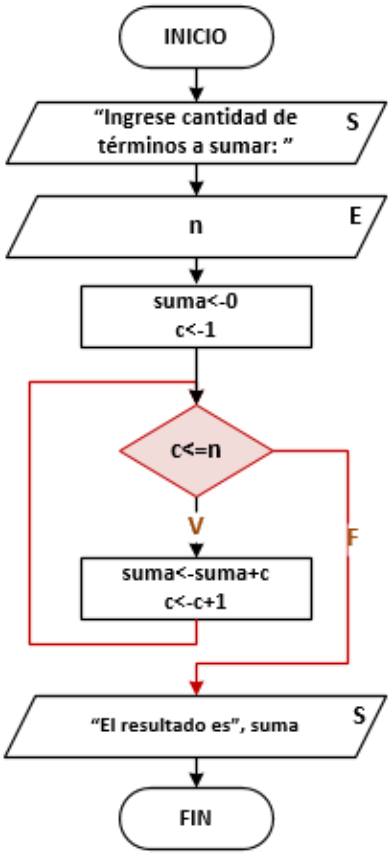
El algoritmo tiene por objetivo **calcular el promedio de un conjunto valores** ingresados por el usuario, lo que implicará ejecutar varias operaciones de lectura y suma para obtener el promedio. En este caso, la cantidad de valores a promediar es desconocida ya que depende de la respuesta del usuario (el ingreso finaliza a petición del usuario). Al desconocer *a priori* la cantidad de datos será necesario contar los valores ingresados mediante la variable *contador*. La estructura REPETIR ejecutará la lectura y suma de valores en tanto la variable *respuesta* sea distinta de 'n' o 'N'; finalizando el bucle cuando el usuario ingrese 'n' o 'N'. Un bucle que utiliza un valor específico para su condición de salida se denomina ***bucle controlado por centinela***.

3. Diseñe un algoritmo (diagrama de flujo y pseudocódigo) que resuelva la siguiente expresión:

$$\sum_{c=1}^{n} c = 1 + 2 + 3 + \cdots + n$$

SUMATORIA: *∑ es una notación matemática utilizada para resumir la suma de n o infinitos términos.*

La variable *c* asumirá los valores del rango especificado por los límites inferior (1) y superior (n) de la sumatoria. Como puede observarse cada término de la suma se corresponde con los valores que tomará la variable *c*.




```
PROGRAMA Sumatoria
VARIABLES
    n, c, suma: ENTERO
INICIO
    ESCRIBIR "Ingrese la cantidad de términos a sumar:"
    LEER n
    suma←0
    c←1
    MIENTRAS c<=n HACER
        suma←suma + c
        c←c + 1
    FIN_MIENTRAS
    ESCRIBIR 'El resultado es:' suma
FIN
```

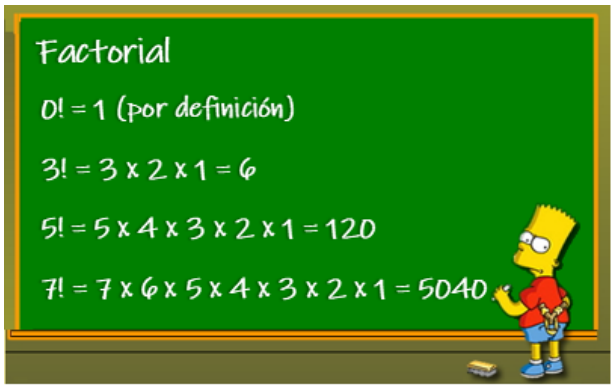
El algoritmo tiene por objetivo **sumar los *n* términos** de la sumatoria especificada. La variable ***c*** funciona como un ***contador*** que controla la cantidad de términos que se sumarán (repeticiones a realizar). En este caso, una estructura repetitiva que utiliza un contador para evaluar su condición de salida se denomina ***bucle controlado por contador.*** La finalización del bucle sucede cuando una variable contador (*c*) alcanza el valor final (*n*). Para almacenar los términos de la sumatoria se utiliza la variable suma (acumulador), cuyo valor inicial será cero (neutro de la operación de suma).

EJERCICIOS A RESOLVER

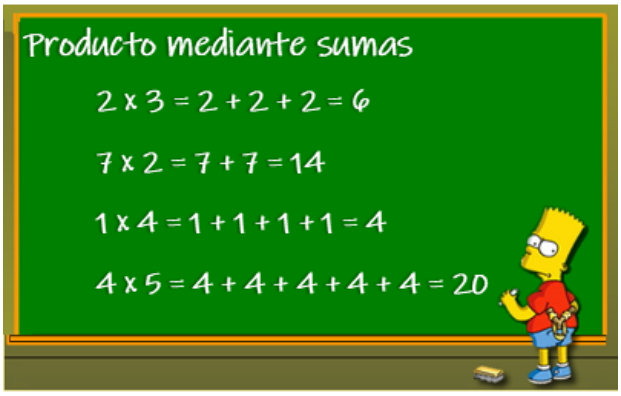
1. Modifique el ejercicio 4 del TP3 añadiendo los controles necesarios para verificar que los valores ingresados sean válidos para el cálculo, obligando al usuario a reingresar éstos si no fueran correctos. ¿Qué estructuras repetitivas puede aplicar? ¿Por qué?
2. Tomando como referencia el ejercicio 5 del TP3, diseñe un algoritmo que permita calcular el perímetro y área de un triángulo, obteniendo ésta última mediante la fórmula de Herón. Incluya los controles necesarios para verificar que los valores ingresados sean válidos para un triángulo, obligando al usuario a reingresar éstos si no fuesen correctos.

3. El factorial de un número entero positivo *n*, denotado como *n!*, se calcula como el producto de los *n* primeros números naturales. Teniendo esto en mente, diseñe un algoritmo (diagrama de flujo y pseudocódigo) que realice este cálculo. Escriba una versión aplicando **cada estructura repetitiva** estudiada.

   Nota: El factorial de *n* es *n!=n x (n-1) x (n-2) x …. x 3 x 2 x 1*. Por ejemplo, el factorial de 5 es 5!=5 x 4 x 3 x 2 x 1 → 5!=120. En particular, el factorial de 0 es **0! = 1**.

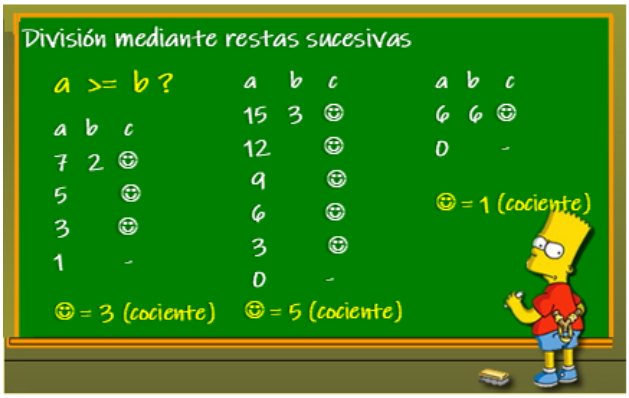


4. Considerando que el producto de dos números enteros positivos, *a* y *b*, puede expresarse como la suma sucesiva de *a*, *b* veces, diseñe un algoritmo (diagrama de flujo y pseudocódigo) que realice el cálculo de producto mediante sumas sucesivas. Desarrolle una versión para **cada estructura repetitiva** estudiada.

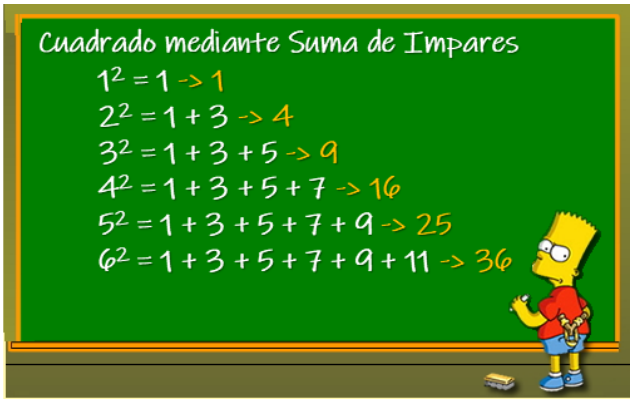


5. Considerando que la división entera de dos números enteros positivos, *a* y *b*, puede expresarse como la resta sucesiva entre *a* y *b* (siempre que *a* sea mayor o igual que *b*), diseñe un algoritmo (diagrama de flujo y pseudocódigo) que calcule el cociente y el resto de la división mediante restas sucesivas. Escriba una versión con estructuras ***MIENTRAS*** y otra con estructuras ***REPETIR***.

   Nota: la *cantidad* de veces que pueda realizar la resta de *a* y *b* (*a >= b*) indica el cociente de la división.



6. Diseñe un algoritmo (diagrama de flujo y pseudocódigo) que calcule el cuadrado de un número entero *N* mediante la suma de los *N* primeros impares. Por ejemplo, el cuadrado de 5 puede calcularse como la suma de los 5 primeros impares 1+3+5+7+9 cuyo resultado es 25. Tenga en cuenta que el cálculo puede realizarse tanto para valores positivos como negativos. Desarrolle una versión para cada estructura repetitiva estudiada y realice la **prueba de escritorio** correspondiente para *N=4, N=5 y N6* (un valor para cada estructura).



7. Sabiendo que el proceso descripto a continuación permite extraer los dígitos de un número, diseñe un algoritmo que sume los dígitos de un número ingresado por el usuario. Escriba 2 versiones del algoritmo, una con estructuras MIENTRAS y otra con estructuras REPETIR.

1984	10			
-4-	198	10		
resto$_1$	-8-	19	10	
	resto$_2$	-9-	1	10
		resto$_3$	-1-	0
			resto$_4$	

La suma de dígitos del número es 22

Proceso

Paso 1: Se divide el número *N* en 10, conservándose el resto obtenido para la suma de dígitos.

Paso 2: Se divide el cociente (entero) obtenido en la división anterior nuevamente por 10, y se conserva el resto para la suma de dígitos.

Paso 3: Se repite el paso 2 hasta que el cociente obtenido sea cero, y se obtiene la suma total de dígitos.

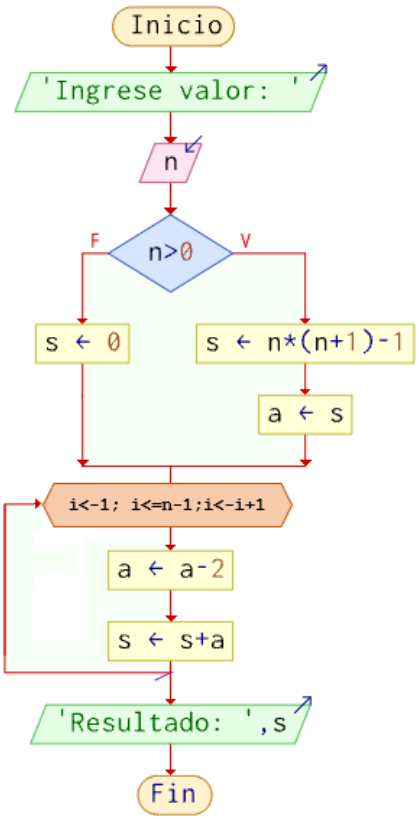
8. Diseñe un algoritmo que permita validar una contraseña ingresada por el usuario, verificando que ésta contenga al menos una letra mayúscula, una letra minúscula, un dígito y un símbolo. Considere que la contraseña podrá componerse de 4 o más caracteres. Incluya los controles necesarios y muestre mensajes que indiquen si la contraseña fue aceptada o se detectaron problemas (identificando cuáles).

9. Diseñe un algoritmo que convierta una cadena de caracteres (compuesta exclusivamente por dígitos) a su equivalente numérico. En el algoritmo propuesto **no utilice ninguna función de conversión de datos**. Además, considere que la cadena puede ser de cualquier longitud.

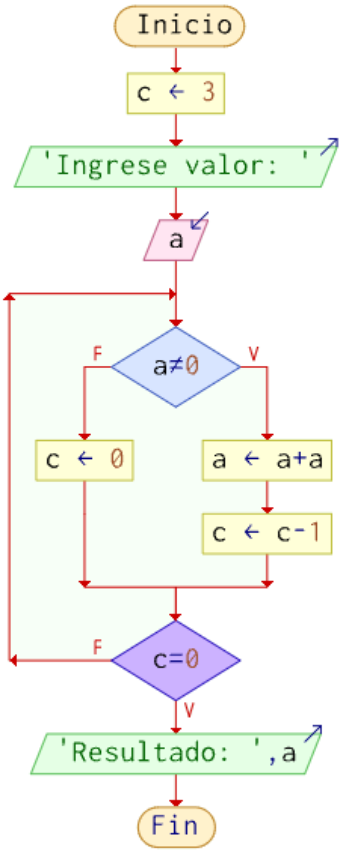
10. Dados los siguientes algoritmos:

```
PROGRAMA engima
VARIABLES
      n, a,res: ENTERO
INICIO
      ESCRIBIR "Ingrese valor: "
      LEER n
      a←0
      res←1
      MIENTRAS n-a>0 HACER
            a←a+1
            res←res*a
      FIN_MIENTRAS
      ESCRIBIR "Resultado: ",res
FIN
```

a. Realice la prueba de escritorio para los valores *n=4, n=5* y *n=6*.
b. Determine el propósito del algoritmo.
c. Dibuje el diagrama de flujo equivalente
d. Escriba otra versión con estructuras *REPETIR*.



a. Realice la prueba de escritorio para los valores *n=2, n=3* y *n=4*.
b. Determine el propósito del algoritmo.
c. Escriba el pseudocódigo equivalente
d. Diseñe otra versión (diagrama de flujo) con estructuras *REPETIR*.



a. Realice la prueba de escritorio para los valores *a=3, a=5* y *a=6*.
b. Determine el propósito del algoritmo.
c. Escriba el pseudocódigo equivalente
d. Diseñe otra versión (diagrama de flujo) con estructuras *MIENTRAS*.

11. Tomando como referencia el ejercicio 4 diseñe un algoritmo que permita calcular el producto, mediante sumas, de 2 valores enteros (positivos o negativos) ingresados por el usuario. Incluya los controles necesarios para calcular correctamente el signo del resultado (regla de los signos).

12. Usando todos tus conocimientos acerca de las estructuras de control, ¿podrías diseñar un juego de adivinanzas numérico?

REGLAS DEL JUEGO:

- El juego consiste en adivinar un número secreto (generado al azar) entre 1 y 99. Para ello, el jugador dispone de 5 intentos y 3 pistas para descubrir el valor secreto. Las pistas son reveladas en los intentos 1, 3 y 5. El juego finaliza cuando el jugador descubre el número secreto y/o se le agotan los intentos.

CONSIDERACIONES PARA LA IMPLEMENTACIÓN

- El número secreto (valor aleatorio) debe generarse utilizando la función azar (ver ayuda PSeInt).
- Conforme se desarrolle el juego se revelarán las siguientes pistas:
  - Pista 1: En el primer intento, se indica si el valor secreto es par o impar.
  - Pista 2: En el tercer intento, se indica el intervalo al que pertenece el valor secreto. Los límites del intervalo no pueden ser mayores a 20 (calculados de forma aleatoria). Por ejemplo, si el valor secreto fuese 45 los límites inferior y superior del intervalo podrían ser 35 (-10) y 65 (+20), respectivamente.
  - Pista 3: En el quinto intento, se indica el intervalo al que pertenece el valor secreto. Los límites del intervalo no pueden ser mayores a 5 (calculados de forma aleatoria). Por ejemplo, si el valor secreto fuese 45 los límites inferior y superior del intervalo podrían ser 41 (-4) y 48 (+3), respectivamente.

  Debe tenerse en cuenta que los límites de los intervalos mostrados para las pistas 2 y 3 no pueden ser inferiores a 1 y mayores a 99. En caso de presentarse esta situación debe indicarse como límite inferior 1 o como límite superior 99.