LABORATORIO DE COMPUTADORAS

TEMA: LÓGICA Y PROGRAMACIÓN

TP 02

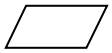
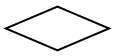
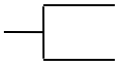
2023

APELLIDO Y NOMBRE: LU:

CARRERA: FECHA:

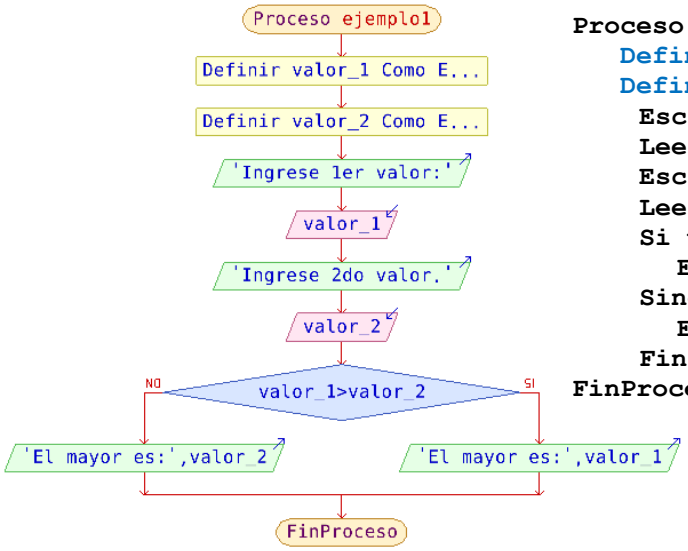
CONCEPTOS

El diagrama de flujo es una herramienta de especificación de algoritmos que permite representar gráficamente (cajas y flechas) los pasos de una solución algorítmica. Esta representación permite visualizar la secuencia de pasos a realizar, los caminos alternativos de acción y el conjunto de acciones que pueden aplicarse de modo repetitivo. En la siguiente tabla se consignan los símbolos más frecuentemente utilizados en diagramas de flujo.

SÍMBOLO	FUNCIÓN
	Terminal, representa el "inicio" y el "fin" de un programa.
	Entrada/salida, representa cualquier tipo de introducción de datos en la memoria desde los periféricos, "entrada", o registro de la información procesada en un periférico, "salida."
	Proceso, representa cualquier tipo de operación que pueda originar cambio de valor, formato o posición de la información almacenada en memoria, operaciones aritméticas, de transferencia, etc.
	Decisión/Decisión Múltiple, indica operaciones lógicas o de comparación entre datos, y en función del resultado determina cuál de los caminos alternativos del programa seguir.
	Conector, permite enlazar 2 partes de un ordinograma a través de un conector en la salida y otro conector en la entrada. Se refiere a la conexión en la misma página del diagrama.
	Indicador de dirección o línea de flujo, señala el sentido de la ejecución de las operaciones.
	Línea conectora, permite conectar 2 símbolos.
	Conector, permite conectar 2 puntos del organigrama situado en páginas diferentes.
	Llamada a subrutina o un proceso predeterminado. Una subrutina es un módulo independiente del programa principal, que recibe una entrada procedente de dicho programa, realiza una tarea determinada y regresa, al terminar, al programa principal.
	Pantalla, se utiliza en ocasiones en lugar del símbolo E/S.
	Impresora, se utiliza en ocasiones en lugar del símbolo E/S.
	Teclado, se utiliza en ocasiones en lugar del símbolo E/S.
	Comentarios, permite añadir comentarios a los símbolos del diagrama de flujo, se pueden dibujar a cualquier lado del símbolo.

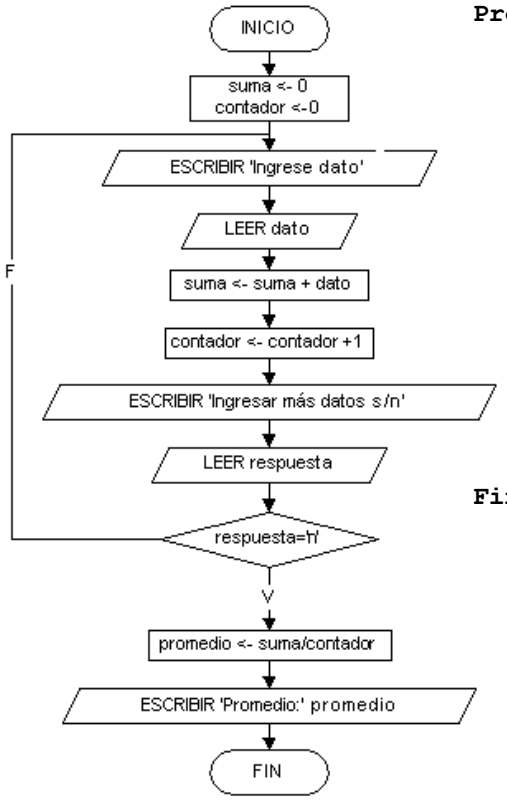
EJEMPLOS

Ejemplo 1: Diseñe un algoritmo (diagrama de flujo) que determine el mayor de dos valores ingresados por el usuario. Codifique en PSeInt.




```
Proceso ejemplo1
   Definir valor_1 como entero;
   Definir valor_2 como entero ;
    Escribir "Ingrese 1er valor:";
    Leer valor_1;
    Escribir "Ingrese 2do valor.";
    Leer valor_2;
    Si valor_1 > valor_2 Entonces
      Escribir "El mayor es:",valor_1;
    Sino
      Escribir "El mayor es:",valor_2;
    FinSi
FinProceso
```

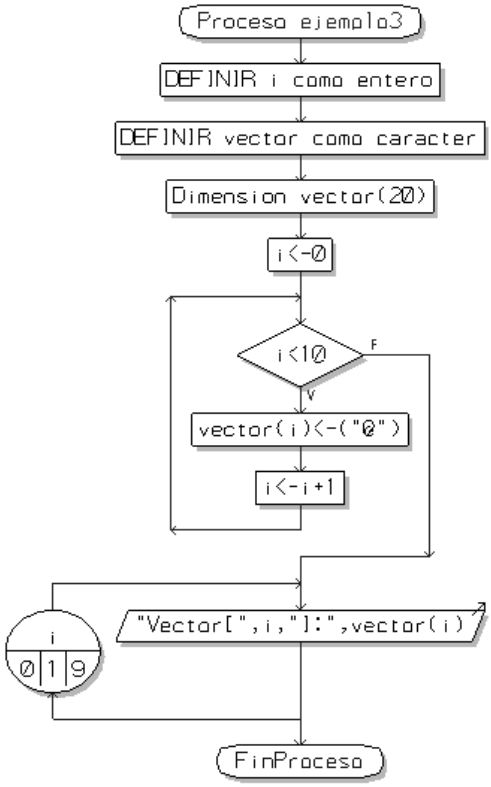
Ejemplo 2: Diseñe un algoritmo (diagrama de flujo) que calcule el promedio de los valores ingresados por el usuario. El ingreso finaliza a pedido del usuario. Codifique en PseInt.




```
Proceso ejemplo2
    Definir suma,dato, promedio como real;
    Definir contador como entero;
    Definir respuesta como caracter;
       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' | respuesta='N');
       promedio<-suma/contador;
       Escribir "Promedio:",promedio;
FinProceso
```

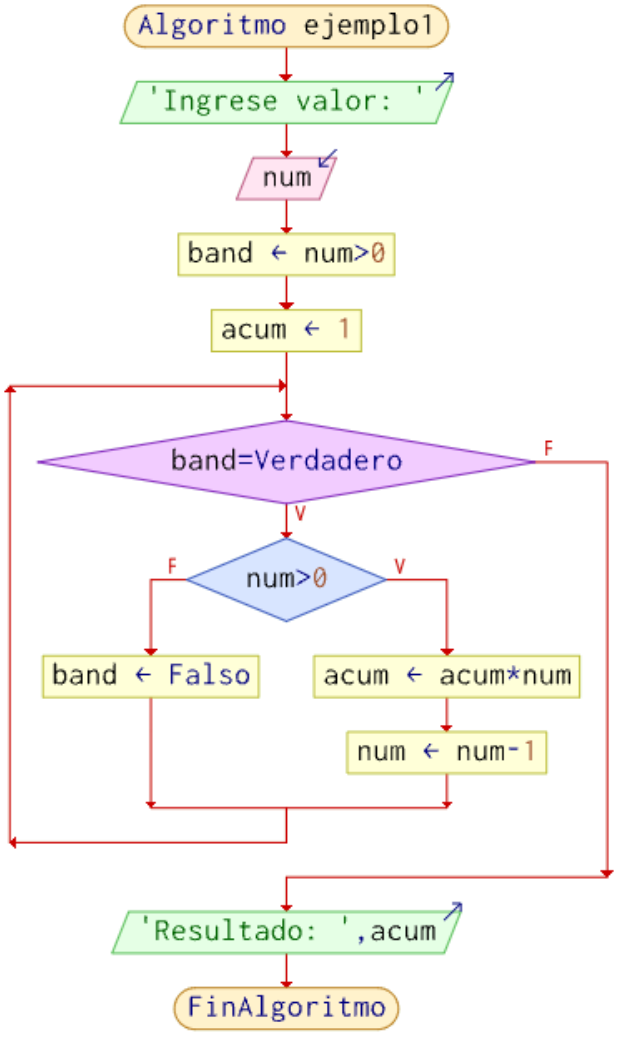
Ejemplo 3: Diseñe un algoritmo (diagrama de flujo) que permita inicializar (con el símbolo @) un arreglo de caracteres de 20 posiciones. Además debe mostrarse por pantalla el contenido del arreglo inicializado. Codifique en PseInt.

```
Proceso ejemplo3
      Definir i como entero;
      Definir vector como caracter;
      Dimension vector[20];
      i<-0;
      Mientras i<10 hacer
            vector[i]<-'@';
            i<-i+1;
      FinMientras
      Para i<-0 Hasta 9 Con Paso 1 Hacer
            Escribir "Vector[",i,"]:",vector[i];
      FinPara
FinProceso
```



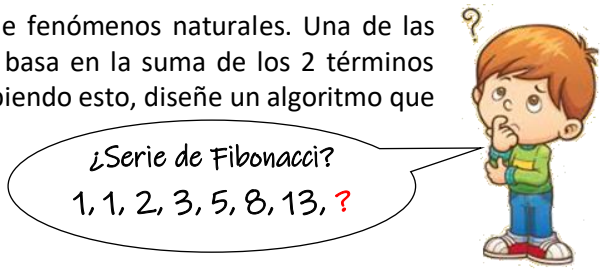
EJERCICIOS

1. Dado el siguiente diagrama de flujo, escriba el pseudocódigo correspondiente y determine su propósito.



2. Diseñe un algoritmo (diagrama de flujo y pseudocódigo) que, dados valores de horas, minutos y segundos, determine si éstos son válidos o no.

3. Las series matemáticas se utilizan para modelar todo tipo de fenómenos naturales. Una de las series más conocidas es la de Fibonacci cuya generación se basa en la suma de los 2 términos precedentes a uno dado, salvo los 2 primeros que valen 1. Sabiendo esto, diseñe un algoritmo que permita generar cualquier término de la serie.



4. Diseñe un algoritmo que permita determinar si un valor ingresado por el usuario es primo o no. Tenga en cuenta que un número natural es primo si únicamente es divisible por la unidad y por sí mismo.

5. Considerando que, la extracción de los dígitos de un número entero puede realizarse siguiendo el proceso descripto a continuación (por ejemplo, para el número 4501), diseñe un algoritmo que calcule la suma de los dígitos de un número ingresado por el usuario.

4501	**10**			
-1-	450	**10**		
resto$_1$	**-0-**	45	**10**	
	resto$_2$	**-5-**	4	**10**
		resto$_3$	**-4-**	0
			resto$_4$	

La suma de dígitos es 10

Proceso

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

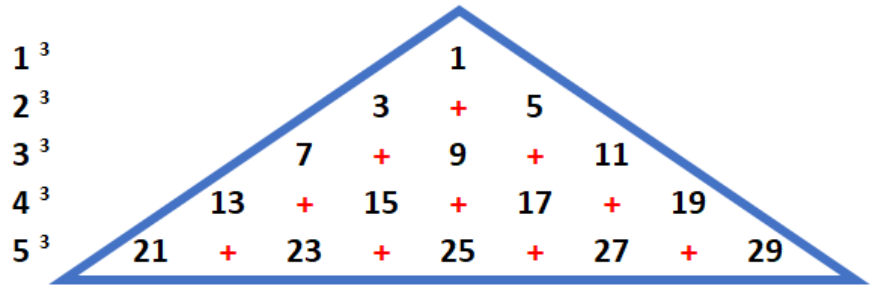
Paso 2: Se divide el cociente (entero) obtenido en la división anterior nuevamente por 10, y se conserva el resto para calcular la suma de dígitos.

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

6. Diseñe un algoritmo que sea capaz de determinar si un número, ingresado por el usuario, tiene todos sus dígitos distintos. Verifique que el número a analizar no tenga más de 5 cifras. Tome como referencia el algoritmo diseñado en el ítem anterior.

7. Sabiendo que el cuadrado de un número entero positivo *N* puede calcularse como las suma de los *N* primeros números impares, por ejemplo, 5^2=1+3+5+7+9 es igual a 25. Diseñe un algoritmo que realice el cálculo del cuadrado utilizando del método descripto.

8. Tomando como referencia el ejercicio anterior, diseño un algoritmo que calcule el cubo de un número entero positivo *N* a partir de la suma de *N* números impares. Para ello, tome en consideración la siguiente pirámide



9. Sabías que un número es divisible por 3 si la suma de sus dígitos es exactamente divisible por 3. Por ejemplo: para saber si el número 3627 es divisible por 3 se suman sus dígitos hasta reducir el valor obtenido a un dígito, es decir, 3+6+2+7=18 y 1+8=9. Si el valor resultante es 3, 6 o 9, el número original es múltiplo de 3.

   Diseñe un algoritmo que sea capaz de determinar si un número es divisible por 3 aplicando el método descripto.

10. El siguiente algoritmo, estructurado de forma modular, permite cargar, listar y sumar los elementos de un arreglo de enteros. Codifíquelo y compruebe su funcionamiento. Luego, agregue 3 módulos: *buscar_valor*, *calcular_maximo_minimo* y *calcular_cubos* (reemplazar los valores positivos del arreglo por su correspondiente cubo).

```
Algoritmo gestion_arreglos
	Definir MAX como entero
	MAX<-10
	Definir num Como entero
	Dimension num[MAX]

	cargar(num,MAX)
	listar(num,MAX)
	Escribir " "
	Escribir "Suma: " suma(num,MAX)
FinAlgoritmo
SubProceso cargar(num,MAX)
	Para i<-1 Hasta MAX Con Paso 1 Hacer
		Escribir "Ingrese: "
		Leer num[i]
	Fin Para
FinSubProceso
Funcion s<-suma(num,MAX)
	Definir i,s como entero
	s<-0
	Para i<-1 Hasta MAX Con Paso 1 Hacer
		s<-s+num[i]
	Fin Para
FinSubProceso
SubProceso listar(num,MAX)
	Definir i como entero
	Escribir "Contenido del Vector"
	Para i<-1 Hasta MAX Con Paso 1 Hacer
		Escribir num[i] " " Sin Saltar
	Fin Para
FinSubProceso
```

11. Dada la siguiente definición de variables determine el propósito del siguiente fragmento de código:

```
Definir MAX como entero
MAX<-5
Definir datos Como caracter
Dimension datos[MAX]
...
	f<-MAX
	ordenado<-FALSO
	Mientras ordenado=FALSO Hacer
		ordenado<-VERDADERO
		Para i<-1 Hasta f-1 Con Paso 1 Hacer
			Si datos[i]>datos[i+1] Entonces
				aux<-datos[i]
				datos[i]<-datos[i+1]
				datos[i+1]<-aux
				ordenado<-FALSO
			Fin Si
		FinPara
		f<-f-1
	FinMientras
```

12. Diseñe un algoritmo que permita agregar valores a un arreglo de enteros, considerando que añadirá sólo un valor al final de los datos cada vez que sea invocada. Además, suponga que el arreglo tendrá una longitud máxima de 15 elementos y que usará una variable auxiliar para indicar la última posición con datos válidos.

13. Diseñe un algoritmo que determine si todos los valores almacenados en un arreglo de caracteres son distintos. Considere que el arreglo tiene una longitud máxima de 20 elementos, y que cuenta con una variable auxiliar que indica la última posición que contiene datos válidos.

14. Tomando como referencia el ejercicio anterior, diseñe un algoritmo determine si un valor entero (de cualquier cantidad de cifras) está formado por dígitos distintos.