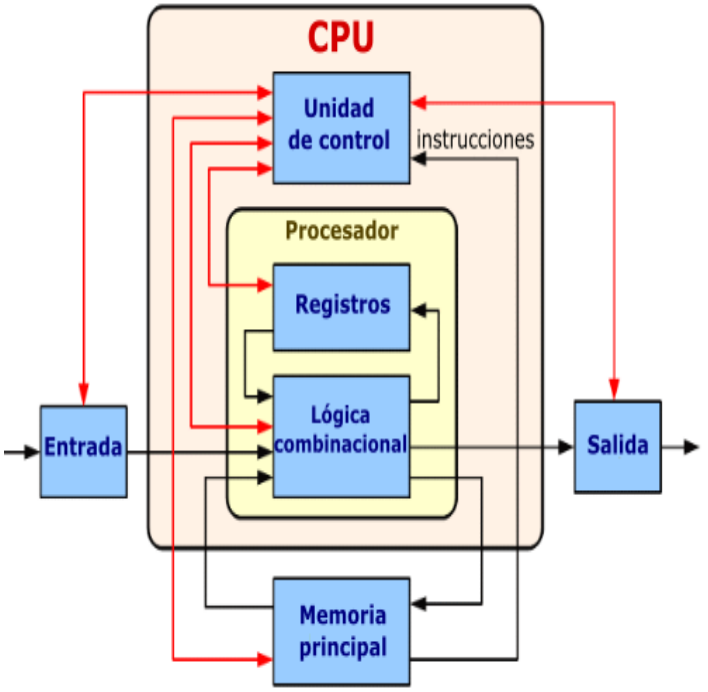
Computador

Es un dispositivo electrónico, capaz de procesar información a gran velocidad y con gran precisión, previa programación correcta del ser humano.

- El computador no piensa
- Procesa instrucciones suministradas mediante algoritmos o programas.
- Solo realiza cálculos matemáticos y lógicos a través de datos y variables.

```
Funcion Expresion()
    Definir x, a,b,c Como Real
    x=0;a=0;b=0;c=0
    Escribir "Ingrese el valor de a: "
    leer a
    Escribir "Ingrese el valor de b: "
    leer b
    c=azar(10)
    Escribir c
    x=((sen(a)+cos(b))*(trunc(a) mod 2))+(raiz(a^3)/(a*b+c))
    Escribir "((sen(a)+cos(b))*(trunc(a) mod 2))+(raiz(a^3)/(a*b+c))=",x
    Escribir "sen=",sen(a)," cos=",cos(b)," mod=",trunc(a) mod 2," raiz= ",ra
FinFuncion
```



Algoritmo

Un algoritmo es una secuencia de pasos bien definidos y ordenados que resuelve un problema o realiza una tarea específica.

Los elementos clave de un algoritmo incluyen:

- la entrada de datos(variables)
- el procesamiento de información mediante operaciones y estructuras de control
- la salida de resultados, el uso de variables y constantes
- la documentación en forma de comentarios.

Estos elementos trabajan juntos para lograr la funcionalidad deseada en un algoritmo. Los algoritmos se expresan a menudo en pseudocódigo o en un lenguaje de programación específico y pueden ser representados visualmente mediante diagramas de flujo.

Variables

Son contenedores que almacenan valores que pueden cambiar a lo largo del programa. Permiten almacenar y manipular datos:
como números enteros, decimales, texto, booleanos y otros tipos de información.

Tabla de operadores relacionales, aritméticos y lógicos usados en las variables

Operador	*Significado*	*Ejemplo*
Relacionales		
>	Mayor que	3>2
<	Menor que	"ABC"<"abc"
=	Igual que	4=3
<=	Menor o igual que	"a"<="b"
>=	Mayor o igual que	4>=5
Logicos		
& ó Y	Conjunción (y).	(7>4) & (2=1) //falso
| ó O	Disyunción (o).	(1=1 | 2=1) //verdadero
~ ó NO	Negación (no).	~(2<5) //falso
Algebraicos		
+	Suma	total <- cant1 + cant2
-	Resta	stock <- disp - venta
*	Multiplicación	area <- base * altura
/	División	porc <- 100 * parte / total
^	Potenciación	sup <- 3.41 * radio ^ 2
% ó MOD	Módulo (resto de la división entera)	resto <- num MOD div

Asignación de variables

Es el proceso de guardar un valor en una variable para que pueda ser utilizado en cálculos posteriores o referenciado en el programa.
Clasificación de las variables. Ejemplos

a=2 // se asigna un valor entero 2 a la variable a
b=5 // se asigna un valor entero 5 a la variable b
sueldo=500.50 // asigna un valor decimal 500.50 a la variable sueldo
nombre = "Daniel" // se asigna en memoria un valor string "Daniel" a la variable nombre
encontrado=true//se asigna el valor true boleano o lógico a la variable encontrado
x = 5+4*a+2/b // se asigna un resultado decimal de esta expresión a la variable x

Clasificación de variables según el Tipo de Datos:

- **Enteros (`int`):** Almacenan números enteros, como -1, 0, 1, 2, etc.
- **Flotantes (`float`):** Almacenan números decimales, como 3.14, 0.001, etc.
- **Cadenas (`str`):** Almacenan texto, como "Hola, mundo".
- **Booleanos (`bool`):** Almacenan valores lógicos, `True` o `False`.

Clasificación de variables según la función de su propósito y cómo se utilizan:

Contadores: son variables que se utilizan para llevar un registro de la cantidad de veces que ocurre un evento o una condición particular en un programa.

- Se incrementan o actualizan en cada iteración de un bucle o cada vez que se cumple una condición específica.
- son útiles para realizar un seguimiento de recuentos, como contar el número de elementos en una lista, contar repeticiones o realizar un seguimiento de eventos particulares**.**

Ejemplo: Contar el número de elementos en una lista de numeros.

Acumuladores: son variables que se utilizan para acumular (sumar o concatenar) valores en un programa.

Se actualizan acumulando valores en cada iteración de un bucle o cada vez que se encuentra un nuevo valor que debe agregarse al acumulado.

Son útiles para calcular totales, sumas, promedios, productos o concatenar cadenas.

Ejemplo: Calcular la suma de todos los elementos en una lista de 5 números

```
Algoritmo Inicio
   acumulador=0
   contador=0
   Mientras contador <5 hacer
      contador=contador+1
      acumulador = acumulador+ contador
   FinMientras
   Escribir contador,acumulador
FinAlgoritmo
```

EJERCICIOS DE LA TAREA 1

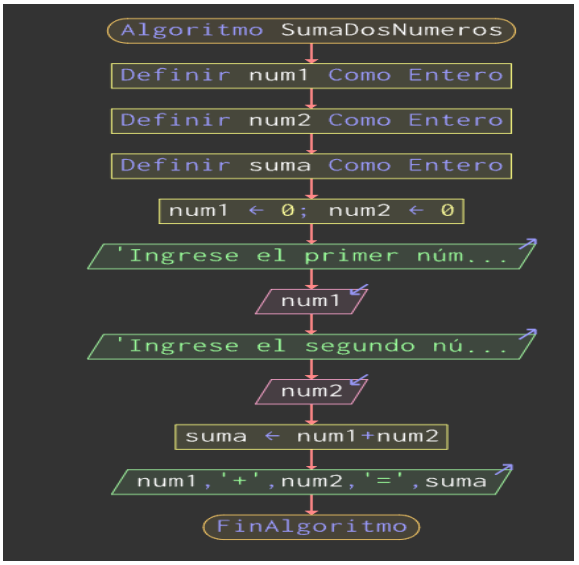
Algoritmos secuenciales

Los algoritmos secuenciales se utilizan en situaciones donde las acciones deben llevarse a cabo en un orden específico, sin necesidad de tomar decisiones condicionales complejas o repetir tareas. Son fundamentales en programación y forman la base para **algoritmos más complejos. Ejemplo:**

Cálculo de la suma de dos números.

En este algoritmo, se pide al usuario que ingrese dos números, se suman y se muestra el resultado.

1. Pedir al usuario que ingrese el primer número.
2. Leer y almacenar el primer número.
3. Pedir al usuario que ingrese el segundo número.
4. Leer y almacenar el segundo número.
5. Sumar los dos números.
6. Mostrar el resultado de la suma.

En Seudocódigo: PseInt	**En Diagrama de flujo**
El seudocódigo es una forma de escribir instrucciones que se asemejan al lenguaje de programación, pero son más simples y fáciles de entender. Ejemplo:	Los diagramas de flujo son representaciones gráficas que utilizan símbolos y flechas para mostrar la secuencia y lógica de un proceso o algoritmo
Algoritmo SumaDosNumeros Definir num1 Como Entero Definir num2 Como Entero Definir suma Como Entero num1=0;num2=0 Escribir "Ingrese el primer número: " Leer num1 Escribir "Ingrese el segundo número: " Leer num2 Suma <- num1 + num2 Escribir num1,"+",num2,"=",suma **FinAlgoritmo**	

Ejercicios de expresiones matemáticas

1. Dado a=3 y b=7, encuentra el valor de y = 2 * a + b - a mod 3.

2. Si a=10 y b=4, calcula el valor de z = a * b + 3 mod a + b.

3. Con a=6 y b=2, determina el valor de w = a - b + 2 * a mod b.

4. Para a=8 y b=5, encuentra el valor de v = 2 * b + a div 2 + 4 * b mod a.

5. Si a=12 y b=9, calcula el valor de u = b - a + 3 * a mod b.

6. (5 + 3 * 2) + 9 > 3 * 5 * 14 % 3

7. 2 *(4 – 10 + 8)/2* 36 *(1/2)

8. 260 / 12 + 54 % 3 – 85 % 7

9. (48 < 2 * 3) | | (2 * 7 < 12)

10. ((8 > 2) | | (932 < 23)) && 4 == 2

Ejercicios algoritmos secuenciales (paso a paso):

11. Suma de dos números: Escribe un programa que tome dos números como entrada y muestre su suma.

12. Área de un triángulo: Pide al usuario que ingrese la base y la altura de un triángulo, luego calcula y muestra su área.

13. Número par o impar: Solicita al usuario que ingrese un número e indica si es par o impar.

14. Calculadora simple: Crea una calculadora que realice operaciones básicas como suma, resta, multiplicación y división, según la elección del usuario.

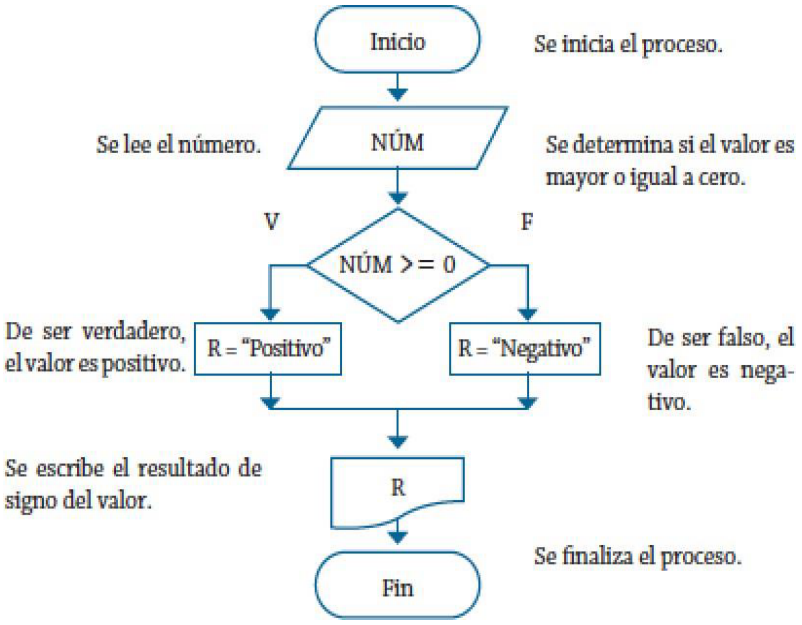
15. Tabla de multiplicar: Pide al usuario un número y muestra su tabla de multiplicar del 1 al 10.

16. Copiar palabra: Escribe un programa que lea dos palabras y concatena en otra variable las dos palabras

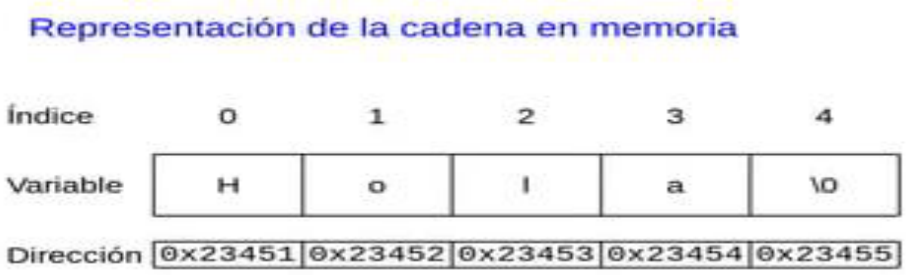
Condiciones

Las condiciones son expresiones o estructuras que permiten tomar decisiones basadas en ciertas circunstancias o situaciones. Estas condiciones son fundamentales para controlar el flujo de un programa y hacer que se comporte de manera diferente en función de las condiciones que se cumplan o no. Ejemplo: Mayor de dos números

Los condicionales más comunes son if-else (si-sino) y switch (selección múltiple).

Seudocódigo	Diagrama de flujo
Funcion mayorDosNumeros() Definir num, R Como Entero Escribir "Ingrese numero" leer num Si num >= 0 Entonces R="Positivo" SiNo R="Positivo" Fin Si Escrubir R FinFuncion	

Strings(cadenas)



Al igual que un arreglo, es una estructura de datos utilizada en programación para almacenar información. Sin embargo, a diferencia de los arreglos que almacenan elementos individuales como números o caracteres, un string es una secuencia de caracteres que representa texto. En resumen, un string es una colección ordenada de caracteres que se utiliza para representar y manipular datos de texto en programación**.**

Ejemplo:

cadena1 = "Hola, "
cadena2 = "mundo!"
resultado = cadena1 + cadena
escribir cadena1 // Hola
escribir cadena1[0] // H
longitudcadena=Longitud(cadena1)
Para indice=0 hasta longitudCadena-1 hacer
    // Escribir cadena[indice]
    Escribir Subcadena (cadena1,indice,indice) // solo se aplica en PseInt
fin para

Ejercicios algoritmos selectivos (con condiciones):

17. Mayor de tres números: Solicita tres números y determina cuál es el mayor de ellos.
18. Edad mínima para votar: Pregunta la edad del usuario y verifica si es elegible para votar (18 años o más).

19. Calculadora de BMI: Crea un programa que calcule el índice de masa corporal (BMI) a partir del peso y la altura del usuario, y luego indique si está en una categoría de peso saludable.

20. Número positivo, negativo o cero: Pide al usuario que ingrese un número y muestra si es positivo, negativo o cero.

21. Año bisiesto: Solicita al usuario un año y determina si es un año bisiesto o no. Un año bisiesto es divisible por 4, pero no por 100, a menos que también sea divisible por 400.
22. Signo zodiacal: Pide al usuario que ingrese su mes y día de nacimiento, luego determina su signo zodiacal. Puedes usar una serie de declaraciones if para comparar las fechas ingresadas con las fechas límite de cada signo zodiacal.

23. Día del mes con respecto a la segunda quincena: Solicita al usuario que ingrese un número de día del mes (por ejemplo, del 1 al 31) y verifica si ese día pertenece a la primera quincena (días 1-15) o a la segunda quincena (días 16-31).
24. Día de la semana: Pide al usuario que ingrese un número del 1 al 7, donde 1 representa el domingo, 2 el lunes, 3 el martes, y así sucesivamente. Luego, utiliza una estructura switch para mostrar el nombre del día de la semana correspondiente al número ingresado.

25. Frases iguales: Escribir un programa que ingrese dos frases e indique si son iguales
26. Calculadora de precio con descuento: Crea un programa que permita a un usuario ingresar el precio de un artículo y un porcentaje de descuento. El programa debe calcular y mostrar el precio final después del descuento.

27. Calculadora de factura con impuestos: Solicita al usuario que ingrese el total de una factura y el porcentaje de impuestos aplicado. Luego, calcula y muestra el monto total a pagar, incluyendo los impuestos.

28. Calculadora de sueldo con aumento: Pide al usuario que ingrese su salario actual y el porcentaje de aumento que recibirá. Calcula y muestra el nuevo salario después del aumento.

29. Calculadora de compra con múltiples artículos: Permite al usuario ingresar el precio y la cantidad de varios artículos que está comprando. Calcula el total de la compra y aplica un descuento del 10% si el total es mayor a cierta cantidad (por ejemplo, $100).

30. Calculadora de impuestos sobre el salario: Solicita al usuario que ingrese su salario anual y calcula el impuesto sobre la renta según las siguientes tasas:
31. Hasta $10,000: 10%
32. Más de $20,000: 15%

33. Descuento por antigüedad en la empresa: Pregunta al usuario cuántos años ha estado trabajando en una empresa y calcula su bono de antigüedad. Si ha trabajado más de 5 años, otorga un bono del 5% sobre su salario.

34. Calculadora de envío con tarifas diferentes: Crea un programa que permita al usuario ingresar la distancia de envío y calcule el costo del envío. Si la distancia es inferior a 50 km, el costo es de $10. Si la distancia es de 50 km o más, el costo es de $20.
35. Calculadora de descuento por lealtad del cliente: Pide al usuario que ingrese el total de sus compras mensuales durante un año. Si el total es superior a $500, aplica un descuento del 10% en la próxima compra.

36. Calculadora de descuento por volumen de compra: Permite al usuario ingresar la cantidad de unidades de un producto que va a comprar y el precio unitario. Aplica descuentos por volumen de compra según las siguientes reglas:
37. 10-50 unidades: 5% de descuento
38. 51-100 unidades: 10% de descuento
39. Más de 100 unidades: 15% de descuento
40. Calculadora de costo de servicio: Pregunta al usuario cuántas horas de servicio necesita y calcula el costo total. Si las horas son más de 10, aplica un descuento del 20%.

Ciclos - For y while

Permiten repetir un bloque de código varias veces. Los más utilizados son el bucle while(mientras) y el bucle for(para).

El bucle **for(para)** se utiliza cuando sabes cuántas veces deseas repetir una acción. Normalmente, se utiliza para iterar sobre una secuencia, como una lista o un rango de números.

```
Desde valor inicial Hasta valor Final
    Proceso
Fin Desde
```

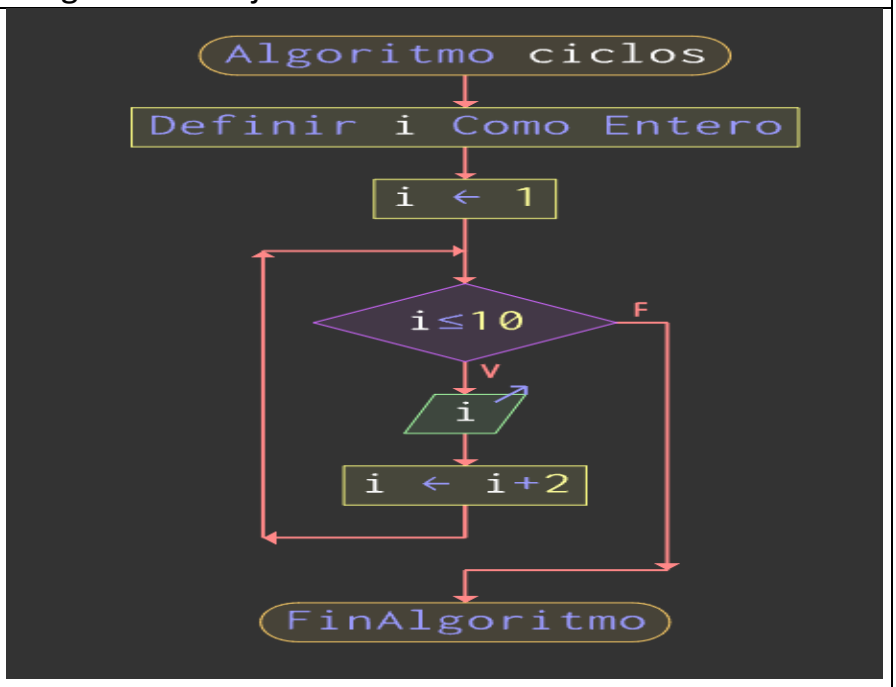
Ejecucion del ciclo for(para)

- La primera vez toma el valor para(inicial) y compara con el valor hasta(final)
- Si el valor inicial no es igual al valor final se ejecutan los procesos dentro del for
- Se regresa al ciclo for y se incrementa la variable inicial y se vuelve a comparar con el valor hasta.
- Este proceso se repite hasta que el valor inicial tome el valor hasta

El bucle **while(mientras)** se utiliza cuando deseas repetir uno o varios procesos mientras se cumpla una condición específica. No sabes de antemano cuántas veces se repetirá el ciclo.

- **Se ejecuta mientras la condición sea verdadera**
- **La variable que interviene en la condición tiene que afectarse en los procesos repetitivos**
- **para que el ciclo se siga ejecutando y en algún**
- **momento la condición deje de cumplirse y finalice**

Ejemplo de seudocódigo y diagrama del mayor de dos números

Seudocódigo	Diagrama de flujo
Algoritmo ciclos definir i Como Entero i=1 Mientras i<=10 Hacer Escribir i i=i+2 Fin Mientras FinAlgoritmo	

Ejercicios:

41. Suma de números pares: Utiliza un bucle for para calcular la suma de los números pares del 1 al 50.

42. Tabla de multiplicar: Utiliza un bucle for para imprimir la tabla de multiplicar de un número ingresado por el usuario del 1 al 12

43. Contador de vocales: Utiliza un bucle while para contar el número de vocales en una palabra ingresada por el usuario.

44. Contador de digitos: Utiliza un bucle for para contar el numero de dígitos en una palabra ingresada por el usuario.

45. Adivina el número: Genera un número aleatorio y pide al usuario que adivine el número. Utiliza un bucle while para repetir la solicitud hasta que adivine correctamente.
46. Contador de Alfabeto: Utiliza un bucle for para contar el número de letras del alfabeto(a..z) en una palabra ingresada por el usuario.

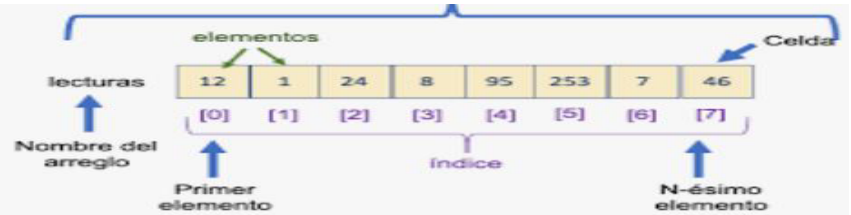
47. Suma de números impares: Utiliza un bucle while para calcular la suma de los números impares del 1 al 100.

48. Contador de caracteres: Escribir un programa que lea una palabra y presenta cuantos caracteres hay en dicha palabra.

49. Suma de números: Pide al usuario que ingrese números enteros positivos uno por uno y utiliza un bucle while para calcular la suma de estos números. El ciclo debe terminar cuando el usuario ingrese un número negativo.

50. Cuenta regresiva: Pide al usuario que ingrese un número entero positivo y utiliza un bucle while para mostrar una cuenta regresiva desde ese número hasta 1.

Arreglos



los arreglos son estructuras de datos utilizadas en programación para almacenar una colección ordenada de elementos del mismo tipo, donde cada elemento se identifica mediante un índice numérico único. Los arreglos son útiles para organizar y acceder a datos de manera eficiente en situaciones donde se requiere almacenar múltiples valores relacionados en una sola variable. Ejemplo:

```
arregloDeNombres = ["Juan", "María", "Carlos", "Ana"]
// Acceder a elementos del arreglo por índice
primerNombre = arregloDeNombres[0]     // "Juan"
segundoNombre = arregloDeNombres[1] // "María"
tercerNombre = arregloDeNombres[2]     // "Carlos"
cuartoNombre = arregloDeNombres[3]    // "Ana"
// Obtener la longitud del arreglo
longitudDelArreglo = longitud(arregloDeNombres) // 4
Para indice=0 hasta longitudDelArreglo-1 hacer
        Escribir arregloDeNombres[indice]
fin para
```

Ejercicios

51. Suma de elementos: Crea un arreglo de números enteros y calcula la suma de todos sus elementos.

52. Promedio de calificaciones: Crea un arreglo de calificaciones (números decimales) y calcula el promedio de las calificaciones.

53. Mayor y menor valor: Encuentra el valor máximo y mínimo en un arreglo de números enteros.

54. Buscar un elemento: Pide al usuario que ingrese un número y verifica si ese número está presente en un arreglo dado.

55. Contar elementos pares: Cuenta cuántos números pares hay en un arreglo de números enteros.

56. Inversión de un arreglo: Invierte el orden de los elementos en un arreglo. Por ejemplo, [1, 2, 3] se convierte en [3, 2, 1].

57. Buscar el índice: Pide al usuario que ingrese un valor y encuentra el índice de ese valor en un arreglo. Si el valor aparece más de una vez, muestra todos los índices.

Funciones

Las funciones son bloques de código con un nombre que realizan tareas específicas. Sirven para reutilizar código, abstraer detalles, organizar el programa y mejorar la modularidad. Debes usar funciones cuando necesitas realizar una tarea en múltiples lugares o deseas estructurar tu código de manera más clara y mantenible.
Ejemplo:

```
Algoritmo EjemploFuncion

   Funcion SumarDosNumeros(Numero1, Numero2)
      // Esta función recibe dos parámetros y devuelve la suma de ambos.
      Devolver Numero1 + Numero2
   FinFuncion

   Escribir "Ingrese el primer número: "
   Leer NumeroA
   Escribir "Ingrese el segundo número: "
   Leer NumeroB
  // Llamamos a la función SumarDosNumeros y guardamos el resultado en la variable Suma
   Suma <- SumarDosNumeros(NumeroA, NumeroB)

   Escribir "La suma de ", NumeroA, " y ", NumeroB, " es ", Suma
FinAlgoritmo
```

Ejercicios

58. Función sin parámetros para saludar.
59. Función con parámetros para sumar dos números.
60. Función con return para multiplicar dos números.
61. Función sin return para determinar si un número es par o impar.
62. Función con parámetros y return para calcular el área de un rectángulo.
63. Función sin parámetros para imprimir tu nombre.
64. Función con return para convertir grados Celsius a Fahrenheit.
65. Función con parámetros para contar un carácter en una frase.
66. Función sin return para imprimir números del 1 al 10.
67. Función con parámetros y return para sumar una lista de números.