



Curso completo

LÓGICA DE PROGRAMACIÓN

Por Sergie Code

PROGRAMACIÓN Y PROGRAMADOR

Programación es el proceso de crear *instrucciones* que le dicen a una **computadora** cómo realizar una **tarea específica**. Estas *instrucciones* se escriben en un lenguaje que la computadora pueda entender, conocido como lenguaje de programación. La **programación** implica la resolución de problemas, el diseño de algoritmos y la traducción de esos algoritmos en un código que la máquina pueda ejecutar.

Ser un **programador** significa ser la persona que diseña, escribe, prueba y mantiene el **código** que permite que los **programas de computadora** funcionen. Los **programadores** no solo crean soluciones técnicas, sino que también aplican *lógica* y *pensamiento crítico* para *resolver problemas* de manera eficiente, traduciendo ideas abstractas en instrucciones precisas que una computadora puede seguir.

```
of
ToString( ),
ToString( String ),
ToString( IFormatProvider ), and
ToString( String, IFormatProvider )
the following output when run in the [en-US]
number is formatted with various combinations
and IFormatProvider.

Provider is not used; the default culture is [
format string: 11876.54
format string: 11.876.54000
format string: 1.187654E+004
format string: 1.18765E+004

CultureInfo object for [nl-NL] is used for the IFormatProvider
format string: 11876.54
'N' format string: 11.876.54000
'E' format string: 1.187654E+004

NumberFormatInfo object with digit group size = 2 and
digit separator = ',' is used for the IFormatProvider:
'N' format string: 1_18_76.54
'E' format string: 1.187654E+004
Press any key to continue . . . -
```





PENSAMIENTO COMPUTACIONAL

Es un enfoque de **resolución de problemas** que implica **descomponer** un **problema complejo** en partes más pequeñas y manejables, identificar **patrones**, abstraer los **aspectos esenciales**, y desarrollar **algoritmos** o instrucciones paso a paso para resolverlo.

Este tipo de **pensamiento** no se limita a la **programación**, sino que puede aplicarse en diversas áreas para analizar y abordar **problemas** de manera *lógica y eficiente*, similar a cómo lo haría una **computadora**.

- Descomposición de problemas
- Identificación de patrones
- Enfoque en lo esencial (Abstracción)
- Desarrollo de algoritmos
- Análisis y mejora de soluciones (Evaluación)

¿QUÉ ES UN PATRÓN?

Un patrón es una repetición identificable y predecible de elementos o características en un conjunto de datos, situaciones o fenómenos.

- **Repetición:** Los patrones son secuencias o características que se repiten de manera constante.
- **Discernibilidad:** Los patrones pueden ser reconocidos y destacados dentro de un conjunto de datos o situaciones.
- **Predictibilidad:** Los patrones permiten anticipar comportamientos o resultados futuros basados en observaciones anteriores.
- **Regularidad:** Los patrones presentan una disposición consistente de sus elementos a lo largo del tiempo o del espacio.
- **Abstracción:** Los patrones se pueden simplificar para representar información de forma más clara y manejable.





¿QUÉ ES UN ALGORITMO?

Un algoritmo es un conjunto de instrucciones precisas y ordenadas que se siguen para realizar una tarea específica o resolver un problema.

- **Secuencia de pasos:** Una serie lógica de pasos que deben seguirse en un orden específico para lograr un objetivo.
- **Precisión:** Cada paso debe ser definido y comprensible para una ejecución precisa y consistente.
- **Finitud:** Los algoritmos deben tener un número finito de pasos, terminando una vez alcanzado el objetivo o después de un número determinado de pasos.
- **Solución de problemas:** Diseñados para resolver problemas específicos.
- **Eficiencia:** Un buen algoritmo no solo logra el resultado deseado, sino que lo hace de manera eficiente, optimizando el uso de recursos como tiempo y memoria.



Sacar una canción en la

GUITARRA

Usando pensamiento
COMPUTACIONAL

Práctica de lo que hemos aprendido



PENSEMOS

Imagina que quieres aprender a tocar una canción en la guitarra y para ello, deseas utilizar el Pensamiento Computacional.

Este enfoque implica desglosar el proceso en pasos más simples, identificar patrones musicales, eliminar detalles innecesarios y seguir un conjunto claro de instrucciones, o algoritmo, para practicar de manera efectiva.





PROCESO

01

Descomposición: Divide el proceso de aprender la canción en pasos más manejables. Escucha la canción para entender su ritmo y melodía. Identifica los acordes o notas principales. Práctica cada acorde o nota individualmente y luego practica la transición entre ellos. Finalmente, intenta tocar la canción completa.

02

Reconocimiento de patrones: Encuentra repeticiones o similitudes en los acordes, progresiones armónicas o ritmo de la canción. Esto te ayudará a entender mejor su estructura y a tocarla de manera más fluida.

03

Abstracción: Enfócate en lo esencial de la canción, como los acordes principales y la estructura general. Ignora detalles menos importantes, como adornos o variaciones de estilo, para simplificar el aprendizaje.

04

Algoritmo: Crea un conjunto ordenado de pasos para aprender la canción. Escucha la canción, identifica los acordes, practica cada uno, trabaja en las transiciones, toca la progresión de acordes en el ritmo y, finalmente, integra la melodía si es necesario para tocarla completa.

05

Evaluación: Revisa el proceso para asegurarte de que sea efectivo y claro. ¿Cubriste todos los aspectos importantes? ¿Los pasos son fáciles de seguir y comprender?

ANALISIS DE UN PROBLEMA

Es el proceso de comprender completamente un **problema** antes de **diseñar una solución**. Implica descomponer el problema en partes más manejables, identificar sus causas subyacentes y establecer un enfoque claro para resolverlo de manera efectiva.

- Identificar y comprender el problema
- Establecer objetivos claros
- Identificar requisitos y restricciones
- Analizar las causas subyacentes
- Explorar soluciones alternativas
- Establecer criterios de éxito



Analisis de un problema

Identificar el problema

Es fundamental entender en qué consiste exactamente el problema. Esto implica identificar todos los detalles relevantes, las entradas y salidas esperadas, así como cualquier restricción o limitación que pueda existir.



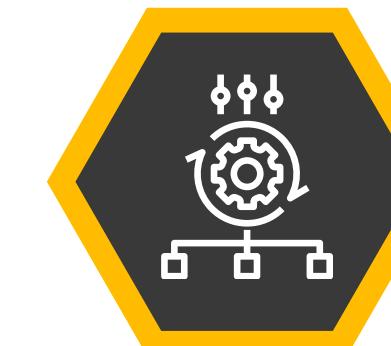
Establecer objetivos

¿Cuál es el propósito final de resolver este problema? Definir claramente los objetivos ayuda a mantener el enfoque y a diseñar una solución efectiva que cumpla con las expectativas.



Requisitos y restricciones

Es importante tener en cuenta las necesidades y expectativas de los usuarios o stakeholders involucrados en el problema. ¿Qué funciones o características son esenciales para ellos?



Causas subyacentes

En ocasiones, un problema puede ser solo un síntoma de una causa subyacente más profunda. Identificar y abordar esta causa raíz es fundamental para evitar que el problema resurja en el futuro.

Soluciones alternativas

Antes de comprometerse con una solución específica, es útil considerar diferentes enfoques y evaluar sus ventajas y desventajas. Esto puede ayudar a encontrar la mejor manera de abordar el problema.

Criterios de éxito

Definir claramente lo que constituirá una solución exitosa es crucial para evaluar el progreso y el rendimiento del programa una vez implementado. Estos criterios pueden incluir medidas de rendimiento, calidad, eficiencia, etc.

Ejemplo

Identificar el problema

Los retrasos en la entrega de paquetes están afectando la satisfacción del cliente y podrían estar generando pérdidas económicas para la empresa.



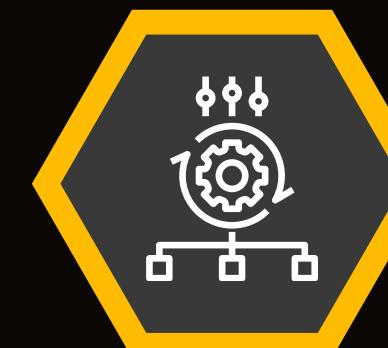
Establecer objetivos

El objetivo principal es reducir los retrasos en la entrega de paquetes a un nivel aceptable para mejorar la satisfacción del cliente y optimizar los costos operativos.



Requisitos y restricciones

Se deben tener en cuenta factores como los plazos de entrega comprometidos, los recursos disponibles, las limitaciones de la infraestructura de transporte, etc.



Causas subyacentes

Se podría descubrir que los retrasos son causados por problemas en la gestión de inventario, deficiencias en las rutas de transporte, falta de coordinación entre los equipos de logística, entre otros posibles factores.

Soluciones alternativas

Se podrían considerar acciones como optimizar los procesos de inventario, mejorar la planificación de rutas, implementar tecnologías de seguimiento en tiempo real, entre otras posibles soluciones.

Criterios de éxito

Se establecerían métricas para medir el éxito, como la reducción del tiempo promedio de entrega, la disminución de quejas de los clientes relacionadas con retrasos, el aumento de la eficiencia operativa, etc.

ESTRUCTURAS DE CONTROL

Las estructuras de control son un conjunto de instrucciones en un programa que permiten alterar el flujo de ejecución de las operaciones, es decir, dictan cómo y cuándo se deben ejecutar las distintas partes del código. Estas estructuras son fundamentales en la programación, ya que sin ellas, los programas seguirían una ejecución secuencial y lineal, sin poder realizar decisiones, repeticiones o desviaciones en el flujo del programa.





Estructuras Condicionales

Las estructuras condicionales permiten que el programa tome decisiones y ejecute ciertas instrucciones solo si se cumplen ciertas condiciones. Esto se logra utilizando sentencias como if, else if (o elif en algunos lenguajes), y else.

Estructuras de Iteración

Los bucles son estructuras de control que permiten repetir una o varias instrucciones múltiples veces, hasta que se cumpla una condición específica. Existen varios tipos de bucles, siendo los más comunes while, for, y do-while.

Estructuras Condicionales

JavaScript

```
let edad = 20;

if (edad < 13) {
    console.log("Eres un niño.");
} else if (edad < 18) {
    console.log("Eres un adolescente.");
} else {
    console.log("Eres un adulto.");
}
```

Python

```
edad = 20

if edad < 13:
    print("Eres un niño.")
elif edad < 18:
    print("Eres un adolescente.")
else:
    print("Eres un adulto.")
```

Java

```
public class Main {
    public static void main(String[] args) {
        int edad = 20;

        if (edad < 13) {
            System.out.println("Eres un niño.");
        } else if (edad < 18) {
            System.out.println("Eres un adolescente.");
        } else {
            System.out.println("Eres un adulto.");
        }
    }
}
```

php

```
<?php
$edad = 20;

if ($edad < 13) {
    echo "Eres un niño.";
} elseif ($edad < 18) {
    echo "Eres un adolescente.";
} else {
    echo "Eres un adulto.";
}
?>
```

Sintaxis:

- JavaScript y PHP utilizan {} para delimitar bloques de código y ; al final de las líneas.
- Python usa la indentación para delimitar bloques, sin necesidad de {}.
- Java también usa {} para bloques de código y requiere el punto y coma ; al final de cada declaración.

Declaración de Variables:

- JavaScript, Python y PHP no requieren especificar el tipo de la variable.
- Java requiere especificar el tipo de la variable (int en este caso).

Método de Salida:

- JavaScript usa console.log para imprimir en la consola.
- Python usa print.
- Java usa System.out.println.
- PHP usa echo.

Estructuras de Iteración

JavaScript

```
for (let i = 1; i <= 5; i++) {  
    console.log(i);  
}
```

Java

```
public class Main {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 5; i++) {  
            System.out.println(i);  
        }  
    }  
}
```

Python

```
for i in range(1, 6):  
    print(i)
```

php

```
<?php  
for ($i = 1; $i <= 5; $i++) {  
    echo $i . "\n";  
}  
?>
```

Sintaxis:

- JavaScript y PHP utilizan la sintaxis `for` (inicialización; condición; incremento) y ; para separar las partes del bucle.
- Python utiliza la función `range(start, stop)` para generar una secuencia de números, y el bucle `for` itera directamente sobre estos números.
- Java utiliza una sintaxis similar a JavaScript y PHP, con `int` para declarar el tipo de la variable de control y `System.out.println` para imprimir.

Declaración de Variables:

- JavaScript y PHP no requieren especificar el tipo de la variable de control (`let` en JavaScript y `$` en PHP).
- Python no necesita especificar el tipo de la variable.
- Java requiere especificar el tipo de la variable (`int` en este caso).

Método de Salida:

- JavaScript utiliza `console.log` para imprimir en la consola.
- Python utiliza `print`.
- Java utiliza `System.out.println`.
- PHP utiliza `echo` y es común añadir "`\n`" para un salto de línea.

Variables

Son contenedores que almacenan datos que pueden cambiar a lo largo del programa. Se declaran, se les asignan valores y se utilizan para realizar operaciones.

Tipo de Datos

Definen la naturaleza de los valores que pueden ser manipulados en un programa. Los tipos más comunes incluyen enteros, decimales, cadenas de texto y booleanos.

Operaciones

Permiten manipular y comparar los datos almacenados en variables para producir nuevos valores, tomar decisiones y controlar el flujo del programa.



Variables y tipos de datos

JavaScript

```
// Número entero
let entero = 42;

// Número flotante
let flotante = 3.14;

// Cadena de texto
let cadena = "Hola, mundo!";

// Booleano
let booleano = true;

// Imprimir los valores
console.log("Entero:", entero);
console.log("Flotante:", flotante);
console.log("Cadena:", cadena);
console.log("Booleano:", booleano);
```

Python

```
# Número entero
entero = 42

# Número flotante
flotante = 3.14

# Cadena de texto
cadena = "Hola, mundo!"

# Booleano
booleano = True

# Imprimir los valores
print("Entero:", entero)
print("Flotante:", flotante)
print("Cadena:", cadena)
print("Booleano:", booleano)
```

Java

```
public class Main {
    public static void main(String[] args) {
        // Número entero
        int entero = 42;

        // Número flotante
        double flotante = 3.14;

        // Cadena de texto
        String cadena = "Hola, mundo!";

        // Booleano
        boolean booleano = true;

        // Imprimir los valores
        System.out.println("Entero: " + entero);
        System.out.println("Flotante: " + flotante);
        System.out.println("Cadena: " + cadena);
        System.out.println("Booleano: " + booleano);
    }
}
```

php

```
<?php
// Número entero
$entero = 42;

// Número flotante
$flotante = 3.14;

// Cadena de texto
$cadena = "Hola, mundo!";

// Booleano
$booleano = true;

// Imprimir los valores
echo "Entero: " . $entero . "\n";
echo "Flotante: " . $flotante . "\n";
echo "Cadena: " . $cadena . "\n";
echo "Booleano: " . ($booleano ? 'true' : 'false') . "\n";
?>
```

Declaración de Variables:

- JavaScript y PHP no requieren especificar el tipo de variable.
- Python también permite la declaración de variables sin tipo explícito.
- Java requiere especificar el tipo de variable (int, double, String, boolean).

Tipos de Datos:

- JavaScript y PHP tienen tipos de datos dinámicos (no necesitan declarar el tipo).
- Python tiene tipado dinámico, similar a JavaScript y PHP, pero utiliza True/False para booleanos en lugar de true/false.
- Java es un lenguaje de tipado estático y requiere especificar el tipo de cada variable.

Método de Salida:

- JavaScript utiliza console.log.
- Python utiliza print.
- Java utiliza System.out.println.
- PHP utiliza echo y requiere la conversión manual de booleanos a cadenas con (\$booleano ? 'true' : 'false') para mostrar true o false.

Funciones

Las funciones o procedimientos son bloques de código que realizan una tarea específica y que pueden ser reutilizados a lo largo del programa. Su uso permite mejorar la organización, la legibilidad y la mantenibilidad del código.

Modularidad

La modularidad se refiere a la práctica de dividir un programa en partes más pequeñas y manejables, llamadas módulos o funciones. Esto tiene varios beneficios:

```
scientist.rb  default.rb
1 | The immutable result of running an
2 | array of candidate Observations.
3 | An Array of candidate Observations.
4 | attr_reader :candidates
5 | The Control Observation to which the rest
6 | attr_reader :control
7 |
8 | An Experiment.
9 | attr_reader :experiment
10 | An Array of observations which didn't match the control.
11 | attr_reader :ignored
12 | An Array of observations which didn't match the control.
13 | attr_reader :mismatched
14 |
15 | An Array of Observations in execution order.
16 | attr_reader :observations
17 |
18 | Internal: Create a new result.
19 |
20 | # experiment - the Experiment this result is for
21 | # observations - an Array of Observations, in execution order
22 | # control - the control Observation
23 |
24 | def initialize(experiment, observations = [], control = nil)
25 |   @experiment = experiment
26 |   @observations = observations
27 |   @control = control
28 |   @candidates = observations - [control]
29 |   evaluate_candidates
30 |
31 |   freeze
32 | end
33 |
34 | # Public: the experiment's context
35 | def context
36 |   experiment.context
37 | end
38 |
39 | # Public: the name of the experiment
40 | def experiment_name
41 |   experiment.name
42 | end
43 |
44 | # Public: was the result a match?
45 | def matched?
46 |   ...
47 |
48 | lib/scientist/result.rb 1:1
```

JavaScript

```
// Definición de la función
function sumar(a, b) {
    return a + b;
}

// Llamada a la función y almacenamiento del resultado
let resultado = sumar(5, 3);

// Imprimir el resultado
console.log("Resultado:", resultado);
```

Python

```
# Definición de la función
def sumar(a, b):
    return a + b

# Llamada a la función y almacenamiento del resultado
resultado = sumar(5, 3)

# Imprimir el resultado
print("Resultado:", resultado)
```

FUNCIONES

Java

```
public class Main {
    // Definición de la función (método)
    public static int sumar(int a, int b) {
        return a + b;
    }

    public static void main(String[] args) {
        // Llamada a la función y almacenamiento del resultado
        int resultado = sumar(5, 3);

        // Imprimir el resultado
        System.out.println("Resultado: " + resultado);
    }
}
```

php

```
<?php
// Definición de la función
function sumar($a, $b) {
    return $a + $b;
}

// Llamada a la función y almacenamiento del resultado
$resultado = sumar(5, 3);

// Imprimir el resultado
echo "Resultado: " . $resultado . "\n";
?>
```

Definición de Función:

- JavaScript usa la palabra clave `function` para definir funciones. Las funciones pueden ser declaradas en cualquier lugar del código.
- Python usa la palabra clave `def` para definir funciones. La definición debe estar correctamente indentada.
- Java define funciones dentro de clases (específicamente, métodos) y requiere que se especifique el tipo de retorno. Además, la definición debe estar dentro de una clase.
- PHP usa la palabra clave `function` similar a JavaScript. Las funciones pueden ser definidas en cualquier lugar del código.

Tipos de Datos y Retorno:

- JavaScript y PHP permiten el uso de tipos de datos dinámicos y no requieren especificar el tipo de retorno.
- Python también tiene tipado dinámico y no requiere especificar el tipo de retorno.
- Java requiere especificar el tipo de retorno de la función y el tipo de cada parámetro.

Llamada a la Función:

- En JavaScript, Python, y PHP, las funciones se llaman directamente con el nombre de la función seguido de paréntesis.
- En Java, las funciones (métodos) se llaman desde un método `main` u otro método en la misma clase.

Estructura de Datos:

La Estructura de Datos en programación es una manera organizada y eficiente de almacenar, manipular y acceder a datos en una computadora. Las estructuras de datos permiten que los programas manejen grandes cantidades de información de manera eficiente y efectiva, facilitando operaciones como búsqueda, inserción, eliminación y modificación de datos.

- **Arreglos (Arrays)**: Colección de elementos del mismo tipo almacenados en ubicaciones de memoria contiguas. Permiten acceso rápido por índice.
- **Listas Enlazadas (Linked Lists)**: Una serie de nodos donde cada nodo contiene un dato y una referencia al siguiente nodo en la secuencia.
- **Pilas (Stacks)**: Colección de elementos que sigue el principio LIFO (Last In, First Out).
- **Colas (Queues)**: Colección de elementos que sigue el principio FIFO (First In, First Out).
- **Árboles (Trees)**: Estructura jerárquica donde cada nodo tiene un valor y referencia a uno o más nodos hijos. Un tipo común es el Árbol Binario.
- **Grafos (Graphs)**: Conjunto de nodos conectados por aristas (edges), útil para representar relaciones complejas entre datos.
- **Hash Tables**: Estructura que almacena pares clave-valor, permitiendo un acceso rápido a los datos a través de una función hash.

ARREGLOS/LISTAS

JavaScript

```
// Definición de un array
let array = [1, 2, 3, 4, 5];

// Acceso a elementos
console.log("Primer elemento:", array[0]); // 1
console.log("Segundo elemento:", array[1]); // 2
```

Python

```
# Definición de una lista (equivalente a array en Python)
array = [1, 2, 3, 4, 5]

# Acceso a elementos
print("Primer elemento:", array[0]) # 1
print("Segundo elemento:", array[1]) # 2
```

Java

```
public class Main {
    public static void main(String[] args) {
        // Definición de un array
        int[] array = {1, 2, 3, 4, 5};

        // Acceso a elementos
        System.out.println("Primer elemento: " + array[0]); // 1
        System.out.println("Segundo elemento: " + array[1]); // 2
    }
}
```

php

```
<?php

// Definición de un array
$array = [1, 2, 3, 4, 5];

// Acceso a elementos
echo "Primer elemento: " . $array[0] . "\n"; // 1
echo "Segundo elemento: " . $array[1] . "\n"; // 2
?>
```

Declaración y Inicialización:

- JavaScript, Python y PHP usan una sintaxis similar para la definición e inicialización de arrays (o listas en Python).
- Java requiere la especificación del tipo de datos del array (int[] en este caso) y los arrays en Java tienen un tamaño fijo.

Acceso a Elementos:

- JavaScript, Python, y PHP usan la misma sintaxis de corchetes [] para acceder a elementos.
- Java también usa [], pero es más rígido en cuanto a tipos y tamaño del array.

Recorrido:

- JavaScript y PHP utilizan un bucle for estándar para recorrer el array.
- Python usa un bucle for simplificado para iterar directamente sobre los elementos.
- Java utiliza un bucle for tradicional con índice o un bucle for-each para recorrer arrays.

Manipulación:

- JavaScript, Python y PHP permiten agregar y eliminar elementos fácilmente con métodos como push, append y array_pop.
- Java requiere la creación de nuevos arrays para agregar o eliminar elementos debido a la naturaleza fija de los arrays.



Paradigmas de programación

Un paradigma de programación es un enfoque o estilo particular de programación que guía y estructura el desarrollo de software.

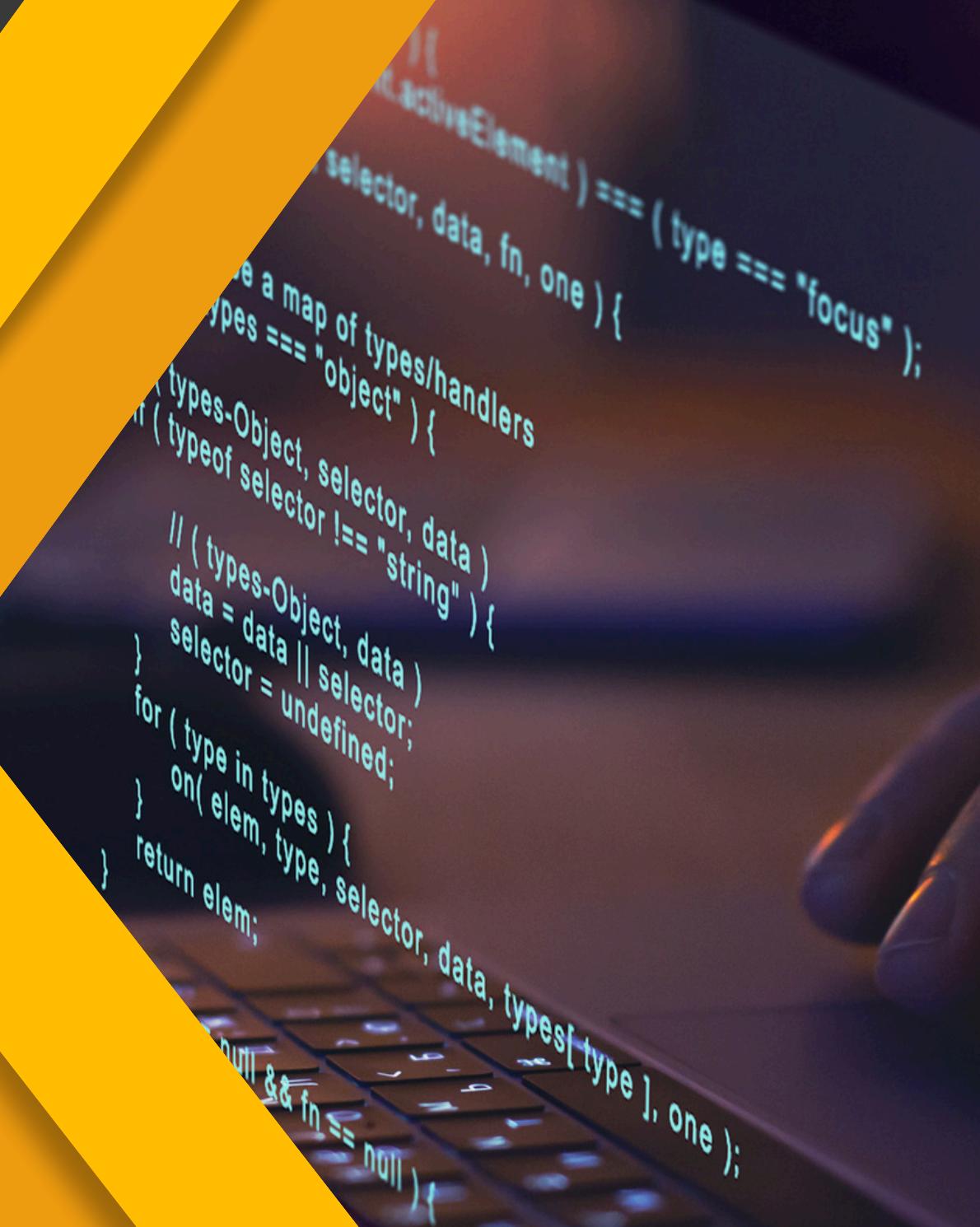
Un paradigma define la manera en que los programadores conceptualizan y organizan los problemas que deben resolver, y dicta cómo se deben estructurar las soluciones en términos de lenguajes de programación y metodologías.

- **Programación Imperativa:** Enfocada en ejecutar instrucciones secuenciales que modifican el estado del programa.
- **Programación Orientada a Objetos (OOP):** Organiza el software en objetos que encapsulan datos y comportamientos. Ejemplos: Java
- **Programación Funcional:** Se basa en funciones puras, evitando el estado mutable y los efectos secundarios.
- **Programación Declarativa:** El programador especifica qué se debe hacer, no cómo hacerlo. Ejemplos: SQL

PSEUDOCÓDIGO

El pseudocódigo es una forma de escribir algoritmos utilizando un lenguaje informal y cercano al lenguaje humano, sin seguir la sintaxis específica de ningún lenguaje de programación en particular. Es una herramienta útil para planificar y diseñar soluciones antes de implementarlas en un código real.

- **Lenguaje informal:** Utiliza un lenguaje que es fácil de entender, parecido al lenguaje natural.
- **Independiente de la sintaxis:** No sigue las reglas de sintaxis de ningún lenguaje de programación específico.
- **Facilita la planificación:** Ayuda a diseñar y organizar el algoritmo antes de escribir el código.
- **Accesible para todos:** Puede ser entendido tanto por programadores como por personas sin conocimientos técnicos.
- **Estructuras lógicas:** Emplea estructuras como condicionales, bucles y asignaciones de manera simple y directa.
- **Enfoque en la lógica:** Se concentra en la lógica y secuencia de pasos más que en detalles técnicos.



Inicio

Definir lista como una colección de números

Definir maximo como el primer número en la lista

Para cada número en lista hacer

Si el número es mayor que maximo entonces

maximo = número

FinSi

FinPara

Imprimir "El número máximo es: " + maximo

Fin

```
Inicio
    Definir n como un número entero
    Definir suma como 0

    Imprimir "Introduce un número entero positivo:"
    Leer n

    Si n <= 0 entonces
        Imprimir "El número debe ser positivo."
    Sino
        Para i desde 1 hasta n hacer
            suma = suma + i
        FinPara

        Imprimir "La suma de los números desde 1 hasta " + n + " es: " + suma
    FinSi
Fin
```

```
t.Fragment>


<div className="container">
  <Title name="our" title=>
    <div className="row">
      <ProductConsumer>
        {(value) => {
          console.log(value);
        }}
      </ProductConsumer>
    </div>
  </div>
</div>
t.Fragment>


```

Diagramas de Flujo

Un diagrama de flujo es una representación gráfica de un proceso o algoritmo, donde se utilizan símbolos y flechas para mostrar la secuencia de pasos y la relación entre ellos. Es una herramienta visual que facilita la comprensión, el análisis y la comunicación de procesos complejos.

- **Visualización clara:** Representa procesos de manera visual, lo que facilita la comprensión.
- **Símbolos estándar:** Usa símbolos estandarizados, como óvalos para inicio/fin, rectángulos para procesos, rombos para decisiones, y flechas para indicar el flujo.
- **Secuencia de pasos:** Muestra el orden en que se deben realizar los pasos de un proceso.
- **Identificación de problemas:** Ayuda a identificar cuellos de botella, redundancias y otros problemas en un proceso.
- **Facilita la comunicación:** Es útil para explicar procesos a personas de diferentes niveles de conocimiento.
- **Aplicaciones variadas:** Se usa en diferentes campos, como programación, ingeniería, administración y más.

SÍMBOLOS POR CONVENCIÓN

Proceso

Documento

Datos o
entrada/
salida

Operación
manual

Operación

Conexión
de
páginas

Decisión

Múltiples
documentos

Database

Retraso

O

Comentario
o
anotación

Terminal

Material

Acceso
directo a
Database

Datos
almacenados

Y

Fecha de
flujo

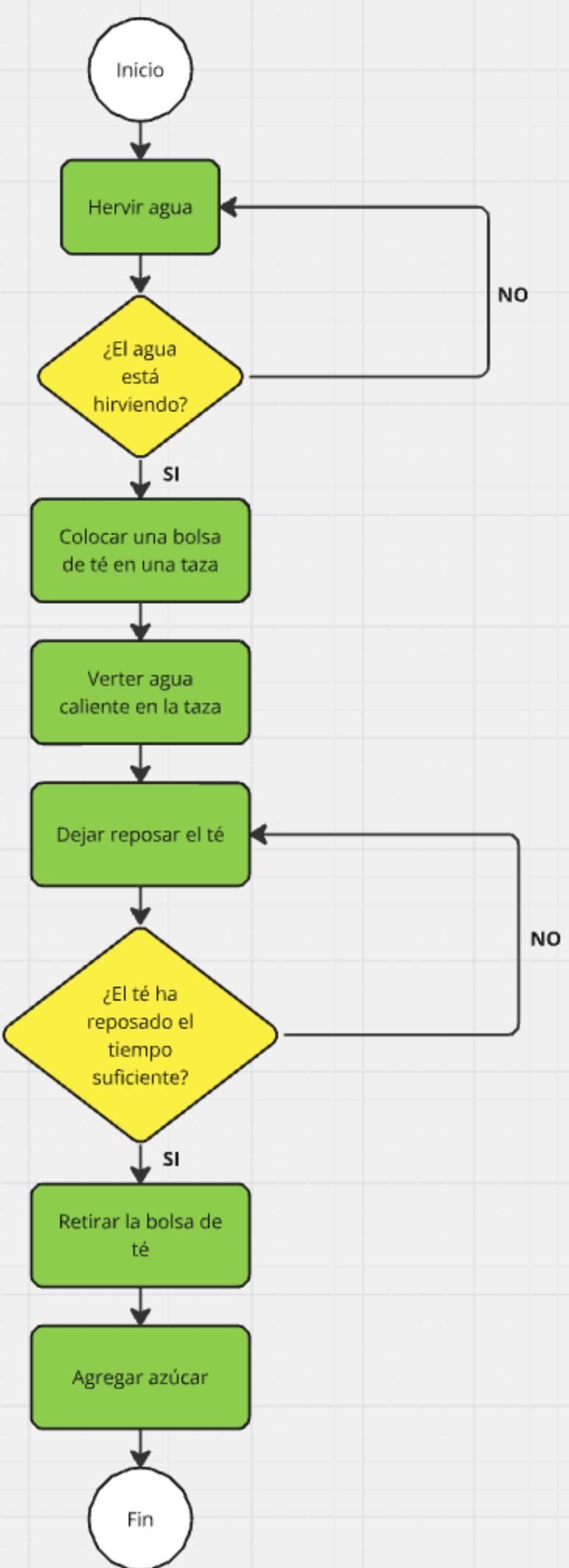
Proceso
predefinido

Preparación

Memoria
principal

Almacena-
miento

Display



ÓVALO

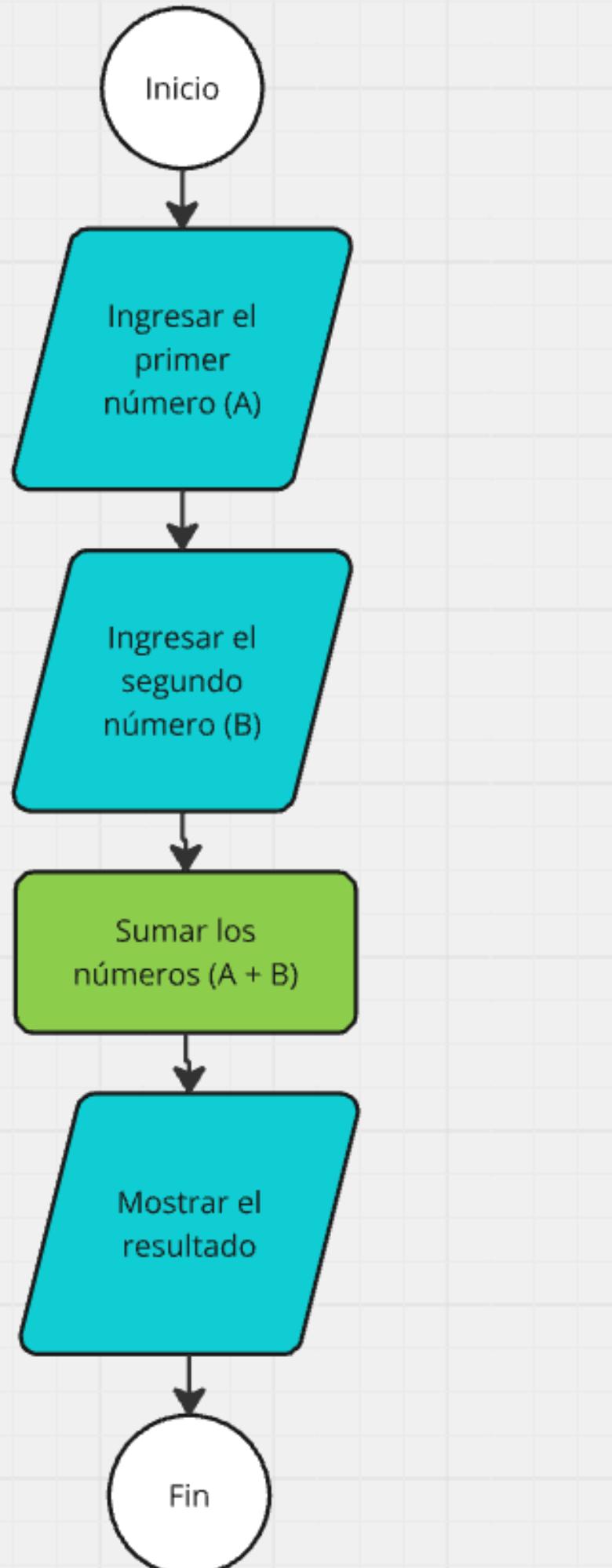
Los óvalos, que a menudo tienen una forma similar a un círculo alargado, se utilizan para indicar el comienzo y el final del proceso.

RECTÁNGULO

Los rectángulos se utilizan para representar pasos en los que se realiza una acción o se lleva a cabo un proceso. Estos pasos son actividades que transforman los datos de entrada en resultados o llevan a cabo tareas específicas.

ROMBO

Los rombos se utilizan para representar puntos en los que se debe tomar una decisión. Estas decisiones se responden con opciones binarias como "Sí" o "No", "Verdadero" o "Falso", que determinan el camino que tomará el flujo del proceso.



ÓVALO

Los óvalos, que a menudo tienen una forma similar a un círculo alargado, se utilizan para indicar el comienzo y el final del proceso.

TRAPECIO

Entrada/Salida. Se utiliza específicamente para representar operaciones de entrada (input) y salida (output), como ingresar datos o mostrar resultados.

RECTÁNGULO

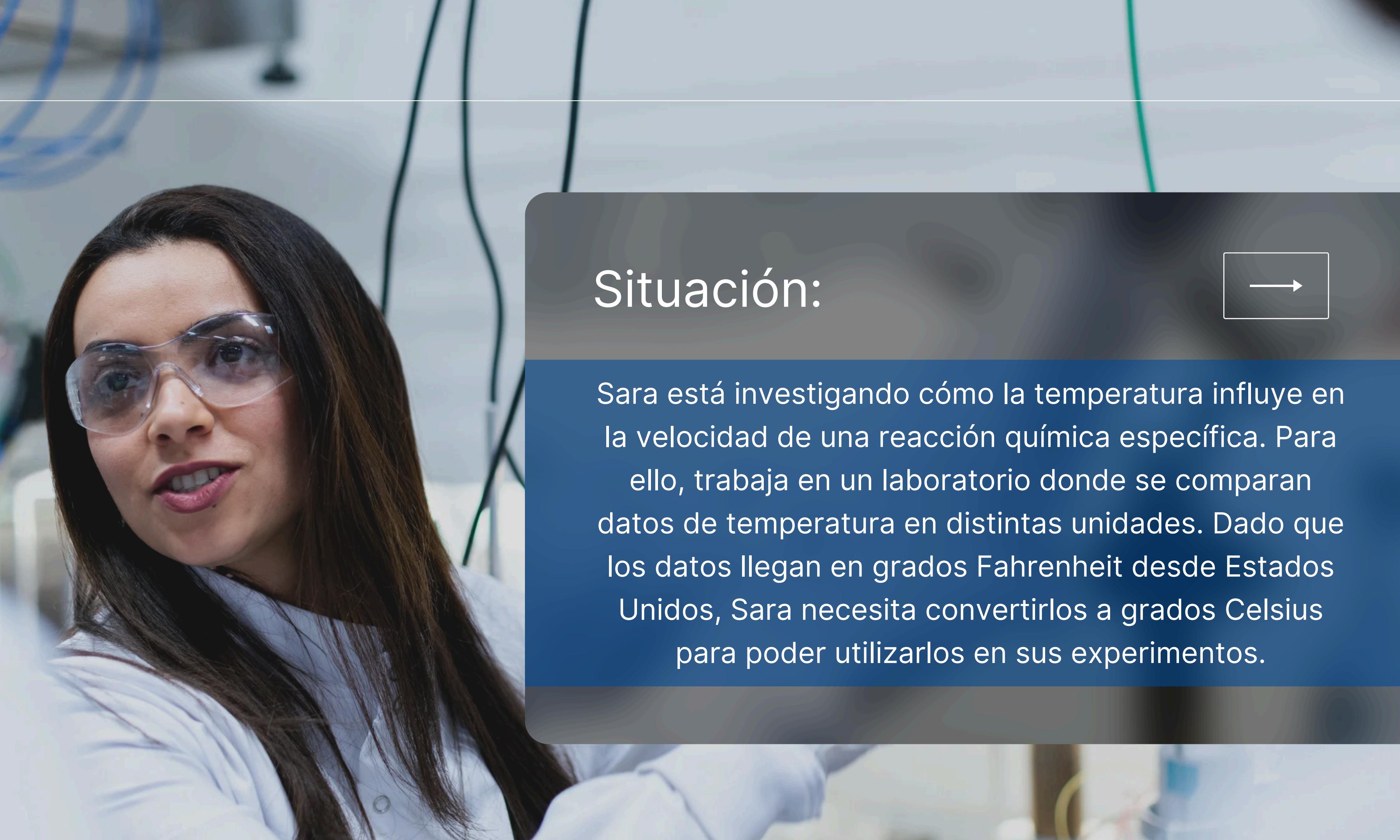
Los rectángulos se utilizan para representar pasos en los que se realiza una acción o se lleva a cabo un proceso. Estos pasos son actividades que transforman los datos de entrada en resultados o llevan a cabo tareas específicas.

+

PROBLEMA CIENTÍFICO

Usaremos todo lo que aprendimos para
ayudar a una científica en su trabajo





Situación:



Sara está investigando cómo la temperatura influye en la velocidad de una reacción química específica. Para ello, trabaja en un laboratorio donde se comparan datos de temperatura en distintas unidades. Dado que los datos llegan en grados Fahrenheit desde Estados Unidos, Sara necesita convertirlos a grados Celsius para poder utilizarlos en sus experimentos.

01

Identificar y comprender el problema: Sara necesita convertir datos de temperatura de grados Fahrenheit a grados Celsius para realizar sus experimentos. La conversión es necesaria para alinear los datos con los otros resultados que ya están en grados Celsius.

02

Establecer objetivos claros: El objetivo principal es convertir con precisión los datos de grados Fahrenheit a grados Celsius y automatizar este proceso para manejar grandes volúmenes de datos de manera eficiente.

03

Identificar requisitos y restricciones: Se requiere precisión en la conversión, automatización del proceso para grandes conjuntos de datos, y compatibilidad con los formatos de análisis existentes. Las restricciones incluyen el formato de datos y el tiempo disponible para el análisis.

04

Analizar las causas subyacentes: El problema surge porque los datos se registran en una unidad diferente a la utilizada en el análisis experimental, requiriendo una conversión para mantener la coherencia en los resultados.

05

Explorar soluciones alternativas: Las posibles soluciones incluyen realizar conversiones manualmente, usar herramientas de hojas de cálculo como Excel, escribir un script para automatizar el proceso, o utilizar software especializado en análisis de datos.

06

Establecer criterios de éxito: El éxito se mide por la precisión de la conversión, la eficiencia en el manejo de grandes volúmenes de datos, la correcta integración de los datos convertidos con los resultados experimentales, y la facilidad de uso del método elegido.