



Recibe una cálida:

¡Bienvenida!

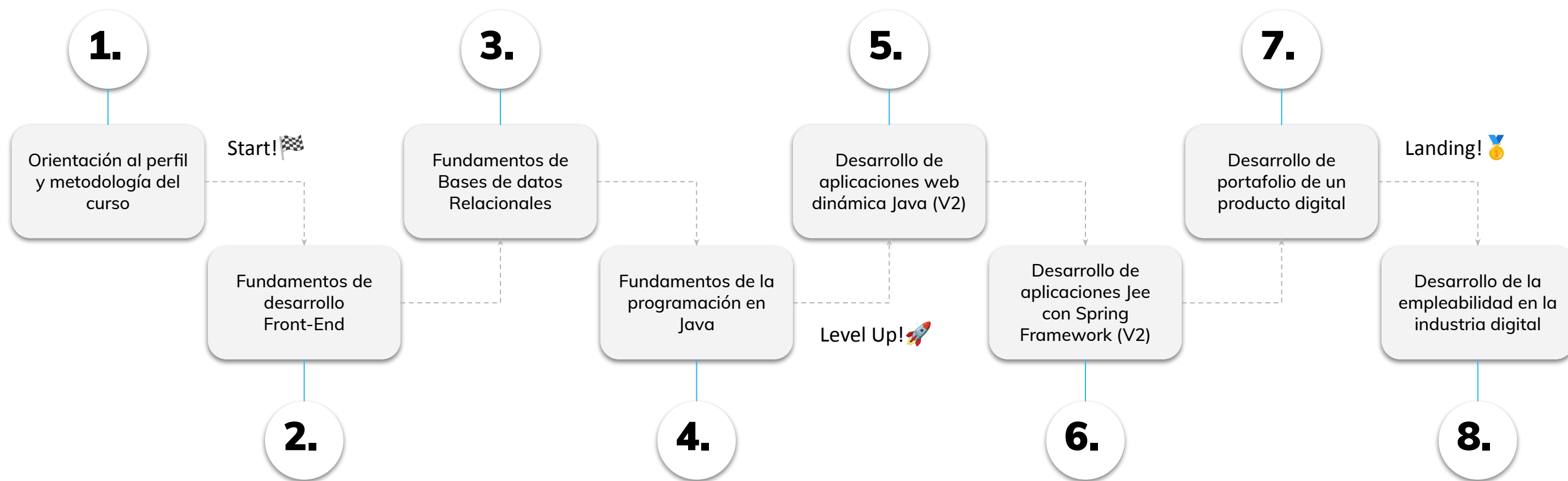
Te estábamos esperando 😊 

» Algoritmos

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



REPASO CLASE ANTERIOR

En la clase anterior trabajamos :

✓ Estructura repetitiva Para

LEARNING PATHWAY

4.2

Start! 🏁

Fundamentos de la programación en Java

Diagramas de flujo y Arreglos

Diagramas de Flujo

Creando un diagrama

Arreglos

Vector

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?



Conocer la importancia y el aplicativo de los diagramas de flujo



Comprender el uso y funcionalidad de los arreglos





Rompehielo



Pasito a pasito: 🙌 Respondan en el chat o levantando la mano

Si tuvieras que representar de manera gráfica (en un cuadro conceptual o cuadro sinóptico) la metodología de estudiar, ¿cómo lo harías?



¿Cuál sería el primer paso?

¿Qué pasaría si no se aprende luego de estudiar? ¿Qué decisiones tomarían?



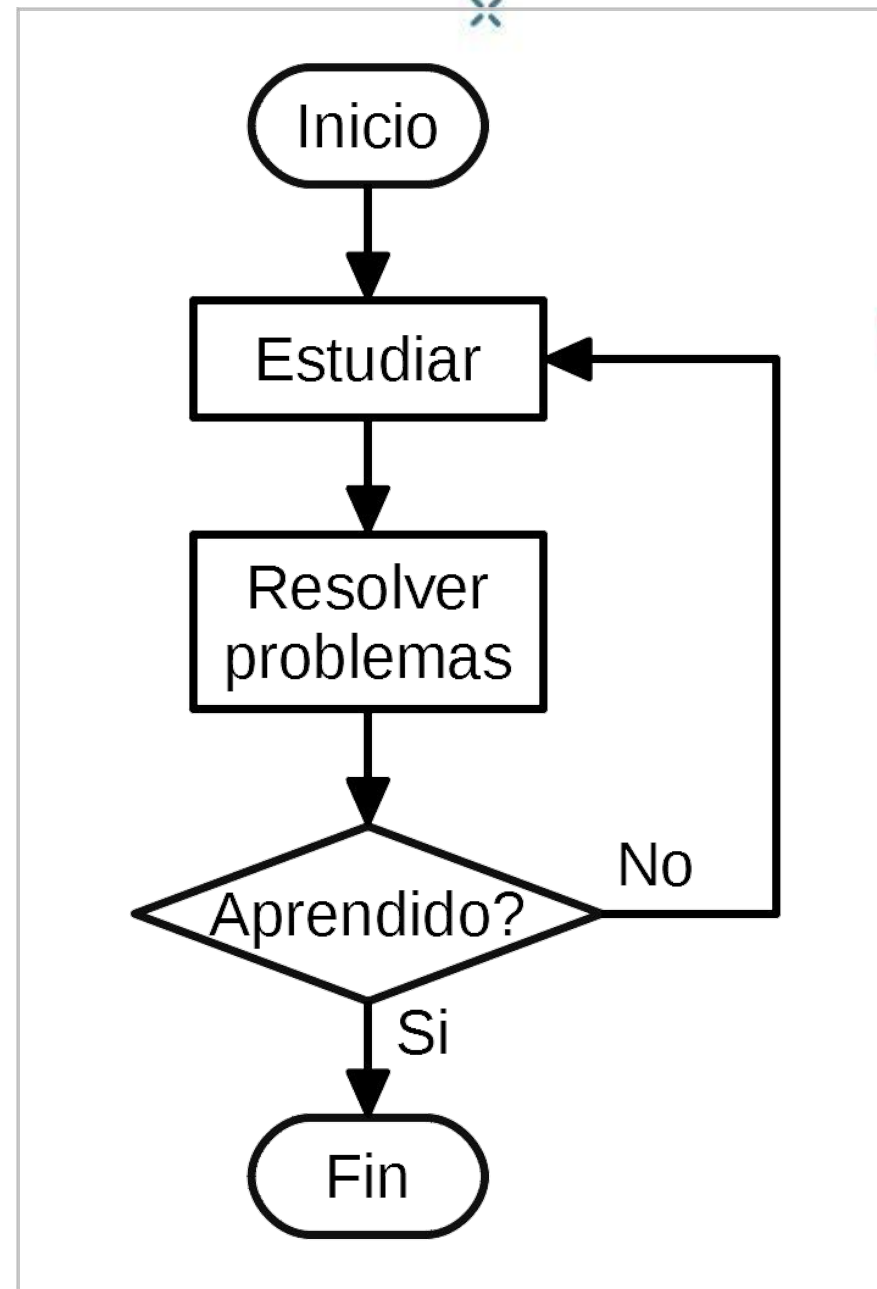


Rompehielo 🧊

Pasito a pasito: 🙌

Aquí tenemos un ejemplo, ¿qué te parece esta forma de representar el paso a paso?

¿A qué te hace acordar?



➤ Diagramas de flujo




Diagrama de flujo

¿Qué es y para qué sirve?:



Un diagrama de flujo es una representación gráfica de un algoritmo. Se utiliza para **representar visualmente los pasos o etapas con las que cuenta un algoritmo**, así como el orden o secuencia de las acciones. Los diagramas de flujo sirven para:

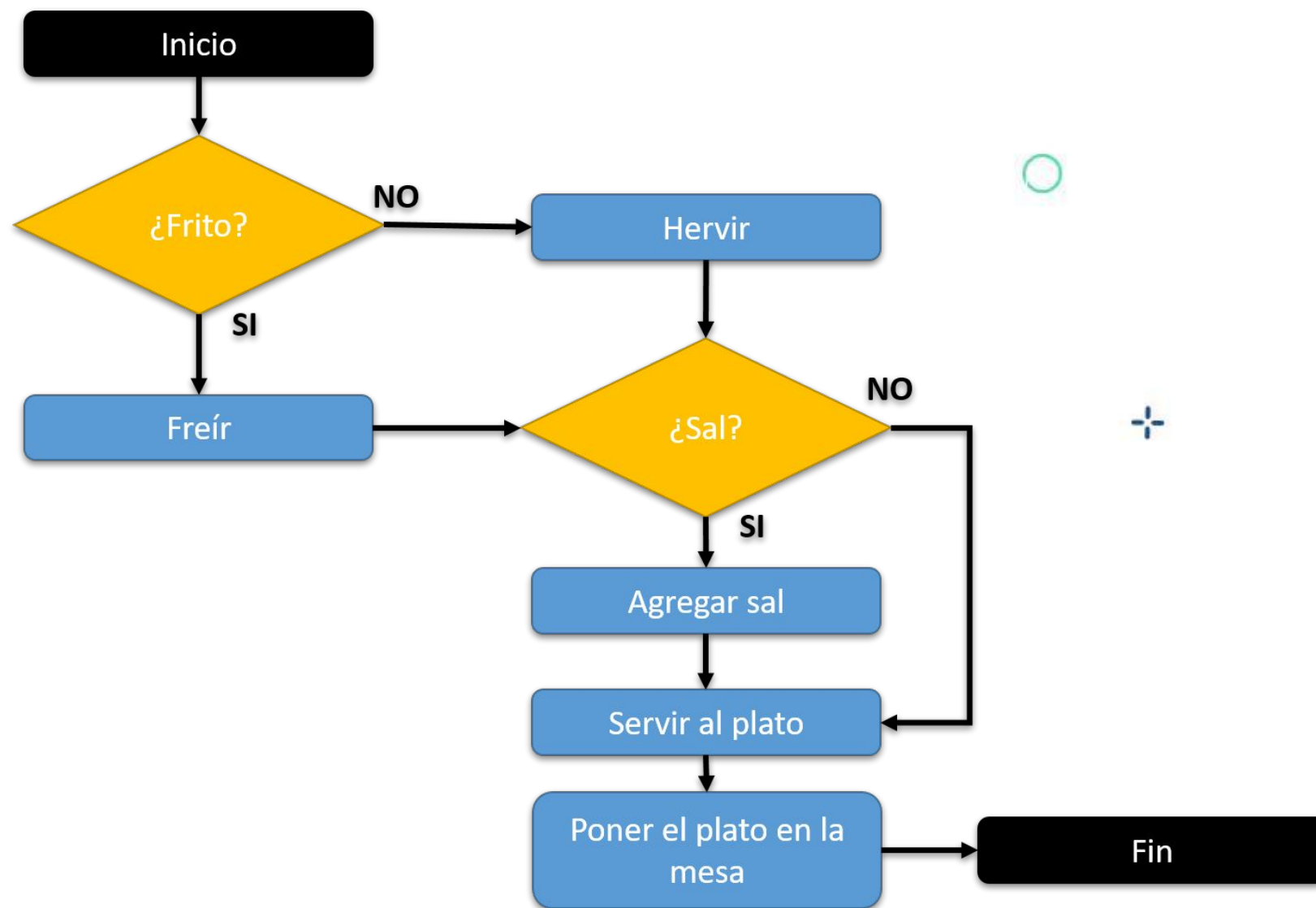


- Describir y documentar procesos de negocio, procedimientos operativos o flujos de trabajo.
- Analizar un proceso actual e identificar oportunidades de mejora.
- Diseñar un proceso nuevo de manera eficiente.
- Comunicar de manera clara y visual un proceso a otras personas.
- Estandarizar la forma en que se realizan ciertas operaciones o tareas.
- Entrenar a nuevos empleados en cómo realizar un proceso.
-  Depurar y optimizar un algoritmo antes de codificarlo.

Diagramas de Flujo

¿Cómo cocinar un huevo?

Aquí podemos ver un ejemplo de la vida cotidiana representado en un diagrama de flujo.














Diagramas de Flujo

Simbología

Además de los símbolos gráficos, se suelen usar palabras clave como "**Inicio**", "**Fin**", "**Sí**", "**No**", para hacer más clara la interpretación del flujo.



Símbolo	Utilidad o significado
	Línea de flujo o flecha, usado para mostrar la continuidad y orden del proceso.
	Usado para indicar el inicio o fin del proceso o subprocesso.
	Indica una actividad del procesos en el diagrama.
	Usado para indicar un decisión de dos opciones (si, no o verdadero, falso).
	Entrada y salida de datos al proceso.
	Documentos del proceso.
	Usado pasa representar un enlace dentro de otra parte del proceso, este lleva una letra o numero en el centro que es con que empieza el enlace.
	Usado para representar demora en la secuencia de una actividad.
	Usado para representar dos partes del diagrama en diferentes paginas.



Ejercicio N° 1

Creando un diagrama



Creando un diagrama

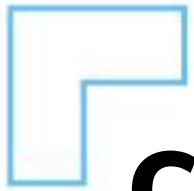
Contexto: 🙌

Vamos a aplicar lo aprendido sobre un ejemplo conocido. Los diagramas de flujo se crean para representar algoritmos. En este caso, el algoritmo será otorgado y sólo deberás centrarte en crear el correspondiente diagrama de flujo teniendo en cuenta la simbología vista anteriormente..

Consigna: 🖋️

Desarrollar un diagrama de flujo que permita leer dos valores distintos, determinar cuál de los dos valores es el **mayor** y mostrarlo por pantalla. En el siguiente slide podrás ver el algoritmo base.

Tiempo 🕒: 15 minutos



Creando un diagrama

Algoritmo:

Inicio

Inicializar variables: $A = 0$, $B = 0$

Solicitar la introducción de dos valores distintos

Leer los dos valores

Asignarlos a las variables A y B

Si $A = B$ Entonces vuelve a leer los valores porque deben ser distintos

Si $A > B$ Entonces

 Escribir A, "Es el mayor"

 De lo contrario: Escribir B, "Es el mayor"

Fin_Si

 Fin

➤ Arreglos



Arreglos



¿Qué son y para qué sirven?:



Los arreglos, también llamados **arrays o vectores**, son estructuras de datos que **permiten almacenar múltiples valores** del mismo tipo **en una sola variable**.

Características principales:



- Los valores se almacenan en posiciones o **índices** numerados de forma secuencial (el primer valor se guarda en la posición 0).
- El número de valores que puede contener el arreglo **se define al declarar la variable**.
- Todos los elementos almacenados deben ser del **mismo tipo de datos** (enteros, cadenas, booleanos, etc).
- Se puede acceder de forma directa a cualquier elemento especificando su índice dentro de corchetes.





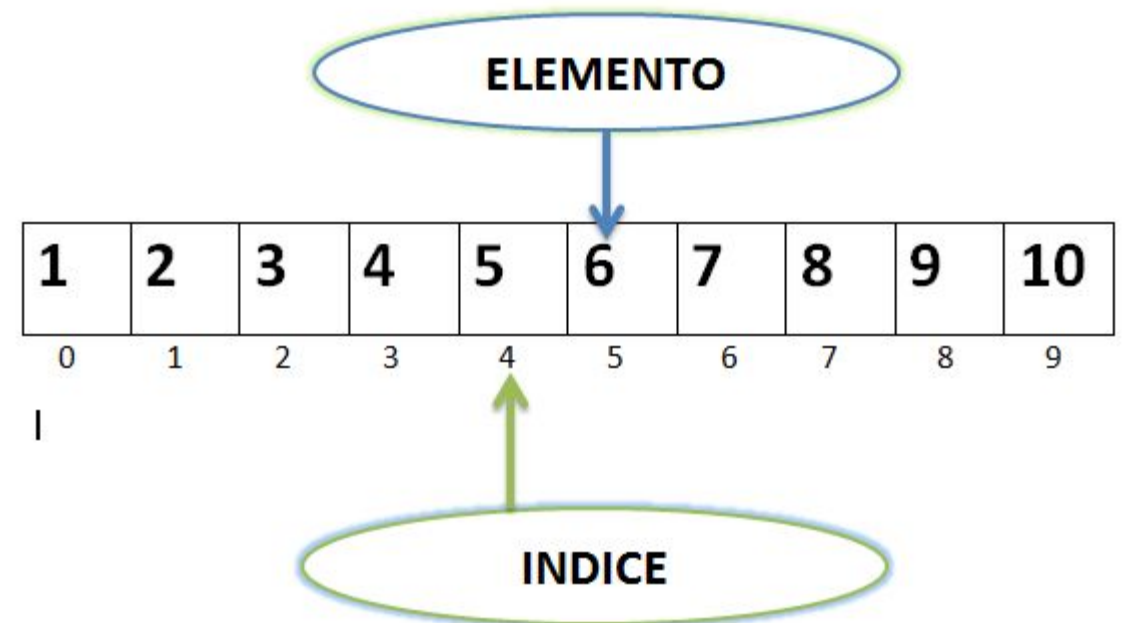
Arreglos



Ventajas:

Los arreglos son estructuras de datos clave en programación que permiten manejar múltiples valores como una colección organizada. Son muy útiles para almacenar y procesar datos de forma eficiente.

- Permite almacenar y organizar múltiples datos en una sola estructura.
- Da un acceso rápido y directo a cualquier elemento especificando su índice.
- El código se simplifica **al poder iterar el arreglo con bucles** en lugar de variables individuales.



Arreglos

UNIDIMENSIONALES: Vectores

- Tiene una sola dimensión, es decir **un solo índice**.
- Los elementos se ordenan en una lista simple.
- Se accede a los elementos con un solo índice.

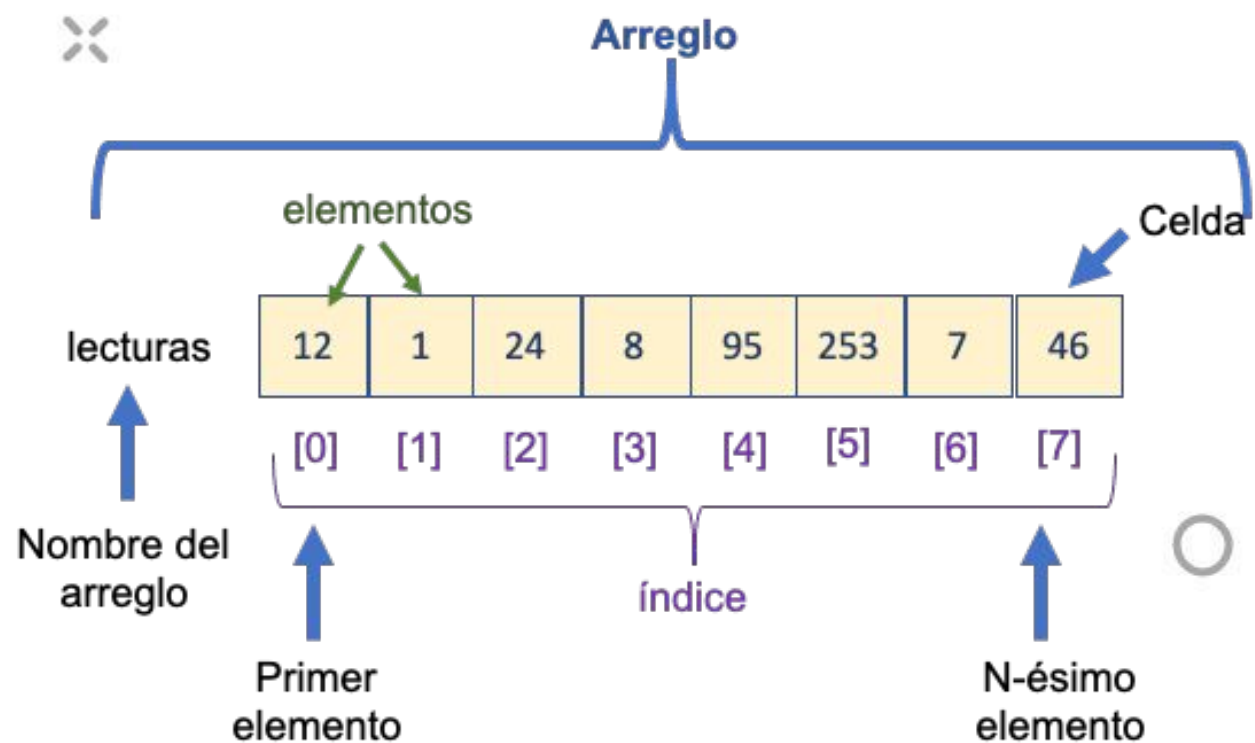
Ejemplo: `numeros[4] = [1, 2, 3, 4, 5]`

BIDIMENSIONALES: Matrices

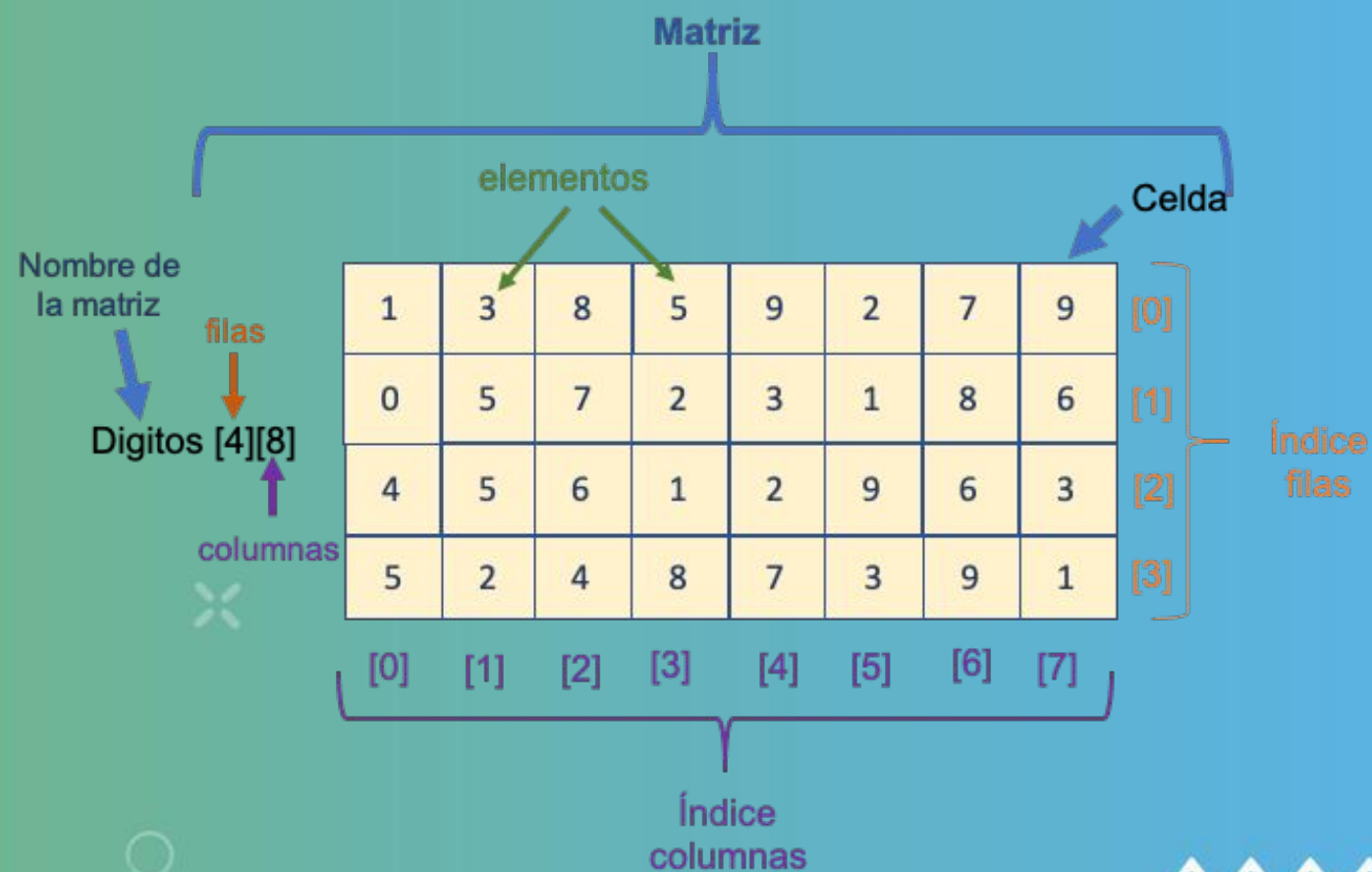
- Tiene dos dimensiones, fila y columna, es decir **dos índices**.
- Los elementos se ordenan en forma de **tabla** o **matriz**.
- Se accede a los elementos con dos índices, **uno para la fila y otro para la columna**.
Ejemplo: `numeros[2,3] = [[1, 2, 3], [4, 5, 6]]`
- Para acceder al elemento 5 se usaría `numeros[1,2]`

Arreglos

UNIDIMENSIONALES: Vector



BIDIMENSIONALES: Matriz





Arreglos Unidimensionales

Declaración: Ejemplo en pseudocódigo

Para declarar un arreglo, se coloca la palabra reservada Dimensión (para aclarar que la variable es del tipo arreglo), luego el nombre de la variable y a continuación el tamaño del arreglo entre corchetes. En este caso, el arreglo será de tipo entero y guardará sólo datos de este tipo.

Definir numeros como entero

Dimension numeros[5]

Para asignarle un valor a cada índice, se puede hacer de manera manual:

```
numeros[0] = 5  
numeros[1] = 10  
numeros[2] = 2  
numeros[3] = 17  
numeros[4] = 8
```

O de manera más automatizada, utilizando un bucle Para.



Arreglos Unidimensionales

Llenar arreglo de 5 elementos con bucle Para

Usar bucles como for para recorrer arreglos es muy útil para inicializarlos, hacer operaciones y más. Los bucles permiten reducir código repetitivo accediendo secuencialmente a los elementos.

```
Para i = 0 hasta 4
    numeros[i] = i * 10
fin para
```

Esto recorrerá el arreglo asignando a cada posición el valor de su índice multiplicado por 10.

- En la primera iteración $i = 0$, se asigna $0 * 10 = 0$
- En la segunda $i = 1$, se asigna $1 * 10 = 10$
- En la tercera $i = 2$, se asigna $2 * 10 = 20$ Y así sucesivamente hasta completar el arreglo.
- Al final el arreglo numeros tendrá los valores: **[0, 10, 20, 30, 40]**



Arreglos Bidimensionales



Declaración: Ejemplo en pseudocódigo

Para declarar una matriz, se coloca la palabra reservada Dimensión (para aclarar que la variable es del tipo arreglo), luego el nombre de la variable y a continuación el tamaño del arreglo entre corchetes. La diferencia principal radica en que la matriz tendrá dos índices.

Definir números como entero

Dimension numeros[2,3] (siendo 2 la referencia de las **filas**, y 3 la referencia de las **columnas**)

Para **asignarle un valor a cada índice**, se recomienda utilizar siempre **bucles Para anidados**.

Se utiliza **un bucle Para por cada dimensión que tenga el arreglo**. En este caso tienen dos dimensiones, por lo que se utilizan **dos bucles Para anidados para recorrer tanto filas como columnas**.





Arreglos Bidimensionales

Llenar arreglo bidimensional con bucles Para anidados.

```
Para i = 0 hasta 1
    Para j = 0 hasta 2
        numeros[i,j] = i + j
    Fin Para
Fin Para
```

Con este bucle for anidado se recorren ambas dimensiones del arreglo:

- La variable **i** recorre las filas (de 0 a 1)
- La variable **j** recorre las columnas (de 0 a 2)
- En cada iteración se asigna a cada elemento la suma de sus índices **i** y **j**.

Al final el arreglo numeros quedará relleno así: $\begin{bmatrix} 0, & 1, & 2 \\ 1, & 2, & 3 \end{bmatrix}$

LIVE CODING

Ejemplo en vivo

Creando arreglos:

- *Vamos a realizar un algoritmo que rellene un vector con los 100 primeros números enteros y los muestre por pantalla en orden descendente.*
- *Luego, vamos a realizar el mismo ejercicio pero rellenando una matriz.*

 **Tiempo: 25 minutos**





Ejercicio N° 2

Vector



Vector

Contexto: 🙌

Vamos a rellenar un vector y a realizar operaciones con sus elementos. Esto es una práctica constante en la programación y te invitamos a comenzar ahora!

Consigna: 🛠️

Desarrollar un algoritmo que calcule la suma de todos los elementos de un vector de tamaño N, con los valores ingresados por el usuario.

Tiempo 🕒: 20 minutos



Paso a paso: ⚙️

1. **Inicializar** variable **suma** en 0
2. Pedir al usuario que ingrese el **tamaño** N del vector
3. **Declarar** el vector de tamaño N
4. Usar un **bucle** para pedir al usuario que ingrese cada elemento del vector
5. Dentro del bucle, ir **sumando cada elemento** ingresado a la variable suma
6. Al finalizar el bucle, la variable **suma** contendrá la suma de todos los elementos

○

¿Alguna consulta?

+



RESUMEN

¿Qué logramos en esta clase?

- ✓ **Comprender el uso e importancia de los diagramas de flujo**
- ✓ **Reconocer la importancia de los arreglos en programación**
- ✓ **Crear y manipular arreglos unidimensionales y bidimensionales**



#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 🙌 🙌 🙌

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
 - a. Material 1 (Foro)
 - b. *Lectura Módulo 4, Lección 2: página 19*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

¡Muchas Gracias!

Nos vemos en la próxima clase 🙌



Momento: ✚

Time-out!

🕒 5 min.

