LÓGICA Y REPRESENTACIÓN I

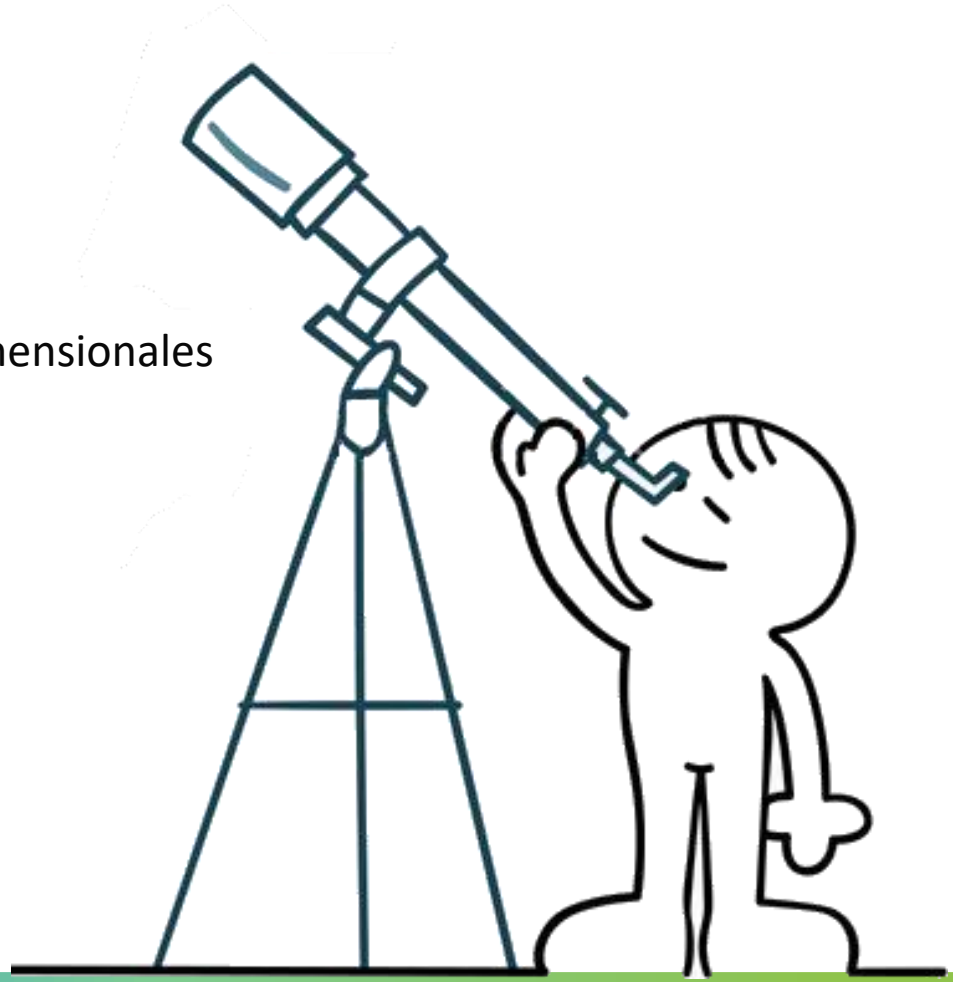
PROGRAMA DE INGENIERÍA DE SISTEMAS

Material desarrollado por **Carlos Andrés Mera Banguero**
https://github.com/carlosmera20/

CONTENIDO

UNIVERSIDAD DE ANTIOQUIA

- **ESTRUCTURAS DE ALMACENAMIENTO ESTÁTICAS - ARREGLOS**
  - Introducción a los arreglos unidimensionales
  - Declaración, almacenamiento y acceso a los datos
  - Ordenamiento de arreglos: burbuja, selección, e inserción
  - Búsqueda lineal y búsqueda binaria
  - Visualización de los datos almacenados en arreglos unidimensionales



Sobre las Estructuras de Datos

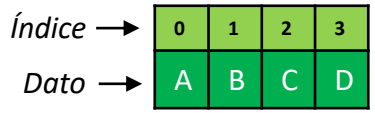
Estructuras de Datos

UNIVERSIDAD DE ANTIOQUIA

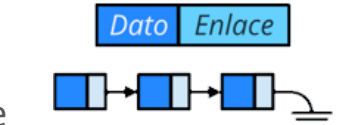
¿Qué son las Estructuras de Datos?

Son “contenedores” que utilizan para almacenar y acceder, de manera organizada, a los datos que se manipulan en un programa. Las Estructuras de Datos además de definir cómo se agrupan y relacionan los datos, también define cuáles son las operaciones que se pueden realizar sobre estos.

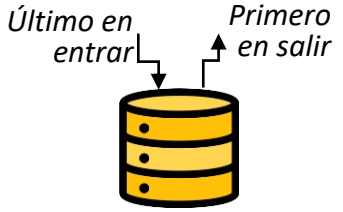
Arreglos: que almacenan una colección de valores del mismo tipo y a los que pueden acceder mediante un índice. Son una estructura estática.



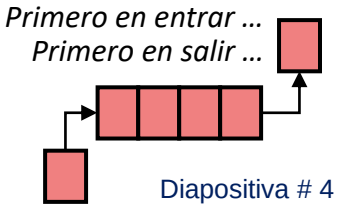
Listas: las cuales almacenan datos que se enlazan unos con otros. Las listas son de tamaño dinámico por lo que permiten agregar, eliminar y modificar datos.



Pilas: almacenan datos siguiendo el principio LIFO (*Last In, First Out*), donde el último elemento en entrar es el primero en salir.



Colas: almacenan datos siguiendo el principio FIFO (*First In, First Out*), donde el primer elemento en entrar es el primero en salir.



¿Qué son los Arreglos?

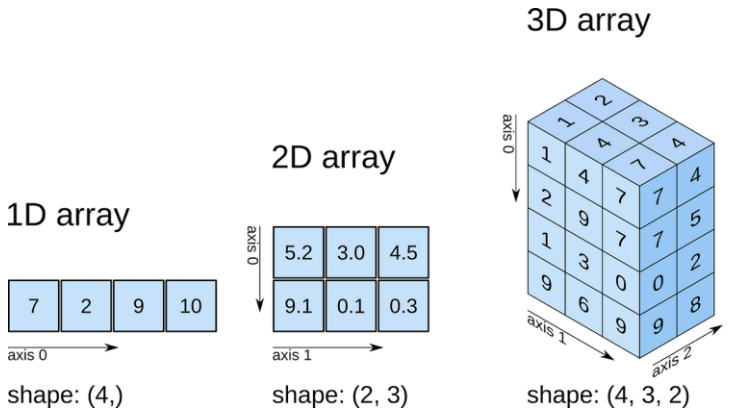
ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA
1803

ARREGLOS

En general, los arreglos son una estructura de datos de tamaño fijo que almacenan múltiples elementos, todos del mismo tipo de dato. En ubicaciones contiguas de la memoria.

Índice →	0	1	2	3
Elementos →	2	8	9	6



¿QUÉ CARACTERIZA A UN ARREGLO?

- **Almacenan valores del mismo tipo de dato**. Los arreglos están diseñados para almacenar elementos del mismo tipo de datos. Esto significa que todos los elementos dentro de un arreglo deben ser del mismo tipo, ya sea entero, flotante, carácter, etc

- **Tienen un tamaño fijo**. Al crear el arreglo se determina el número máximo de elementos que va a almacenar. Los arreglos son cajas con compartimentos, donde cada compartimento contiene un valor (o elemento).

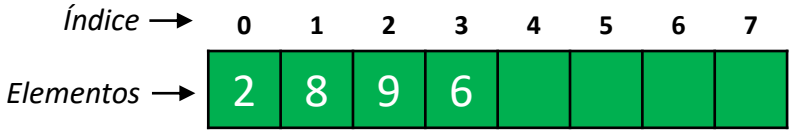
- **Cada elemento del arreglo está numerado**, y a este número se le denomina índice. El índice es el que nos permite acceder a cada elemento almacenado en el arreglo.

ARREGLOS UNIDIMENSIONALES

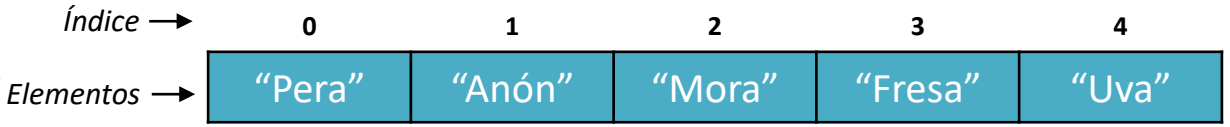
UNIVERSIDAD DE ANTIOQUIA
1803

ARREGLOS UNIDIMENSIONALES - EJEMPLOS

Este arreglo de números enteros que tiene un tamaño de 8. Las posiciones del arreglo se enumeran de 0 a 7 y en este caso sólo hay almacenados valore en 4 posiciones.



Este arreglo de cadenas tiene un tamaño de 5. Las posiciones del arreglo se enumeran de 0 a 4. En este caso todas las posiciones están almacenando valores.



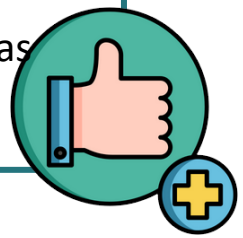
Arreglos Unidimensionales

UNIVERSIDAD DE ANTIOQUIA

Ventajas y Desventajas de los Arreglos

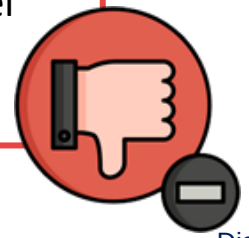
Ventajas

- Por cómo se almacenan en la RAM, el acceso a los elementos, de acuerdo con su posición es más rápido que usar múltiples variables para el mismo propósito.
- Permiten almacenar múltiples datos del mismo tipo bajo el nombre de una misma variable.
- Los arreglos se pueden usar como base para la construcción de otras estructuras de datos como las listas enlazadas y las pilas.

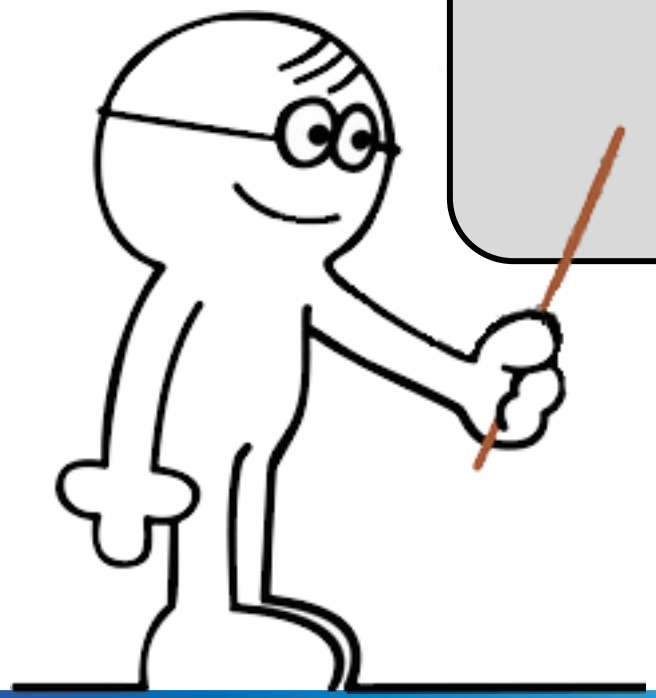


Desventajas

- Tienen un tamaño fijo, así que una vez se crean no se puede aumentar o disminuir el número de datos que almacenan.
- También suele ser una desventaja el que no permitan almacenar valores de múltiples tipos de datos al tiempo.
- La asignación de memoria para un arreglo puede resultar problemática, especialmente en sistemas con memoria limitada. Si el tamaño del arreglo es muy grande, el S.O se puede quedarse sin memoria.



¿Cómo se declara un Arreglo Unidimensional?



ARREGLOS UNIDIMENSIONALES

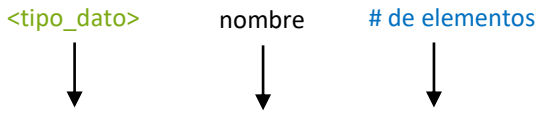
UNIVERSIDAD DE ANTIOQUIA

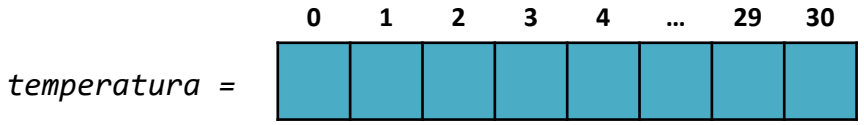
- **DECLARACIÓN DE UN ARREGLO:** existen dos formas de declarar un arreglo unidimensional.

La primera forma consiste en declarar el arreglo especificando su tamaño, el cual es fijo. La forma general para crear un arreglo así es la siguiente:

```
<tipo_dato> nombre_arreglo (numero_elementos)
```

- *Por Ejemplo:* si vamos a crear un arreglo para almacenar la temperatura de los 31 días de marzo, la forma de hacerlo sería:

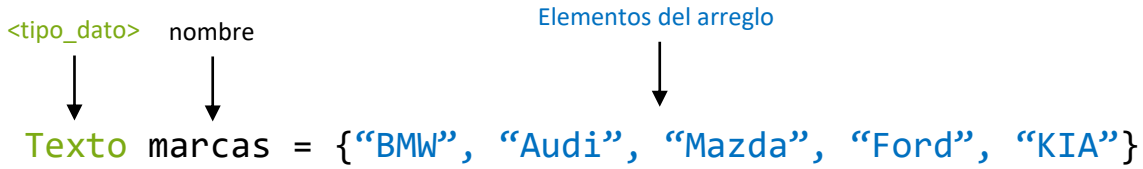


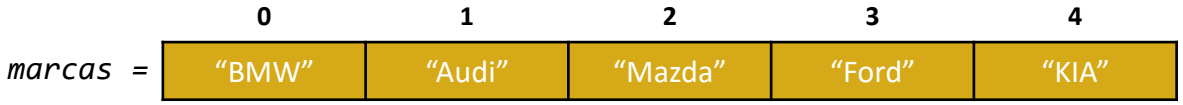


La segunda forma de crear un arreglo consiste en definirlo indicando cuáles son sus valores iniciales, lo que también define su tamaño. La forma de hacerlo es así:

```
<tipo_dato> nombre_arreglo = {val_1, val_2, …, val_n}
```

- *Por Ejemplo:* creemos un arreglo que almacene los nombres de 5 marcas de autos:





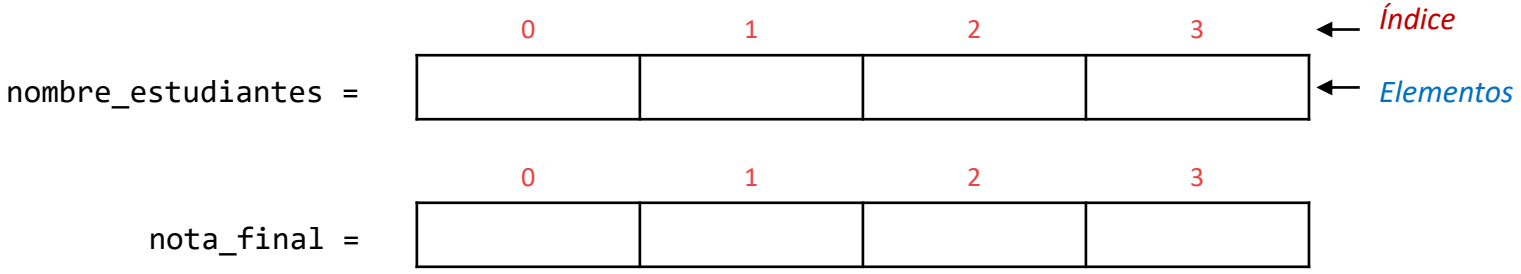
ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

- **DECLARACIÓN DE UN ARREGLO:** veamos otro ejemplo, vamos a crear dos arreglos, uno para almacenar los nombres y otro para almacenar las notas finales de Lógica de 4 estudiantes. En este caso ambos arreglos se crean vacíos:

```
Texto nombre_estudiantes(4)
  ↑          ↑          ↑
<tipo_dato>  <nombre>  # de elementos
  ↓          ↓          ↓
Real       nota_final(4)
```

Imaginariamente, podemos pensar estos arreglos como:



ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

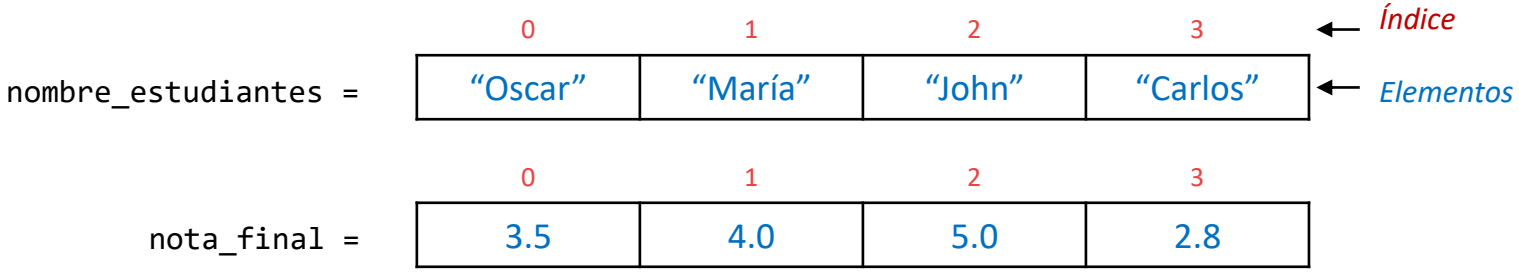
- **DECLARACIÓN DE UN ARREGLO:** Ahora vamos a crearlos usando valores por defecto:

```
Texto nombre_estudiantes = {“Oscar”, “María”, “Jhon”, “Carlos”}
```

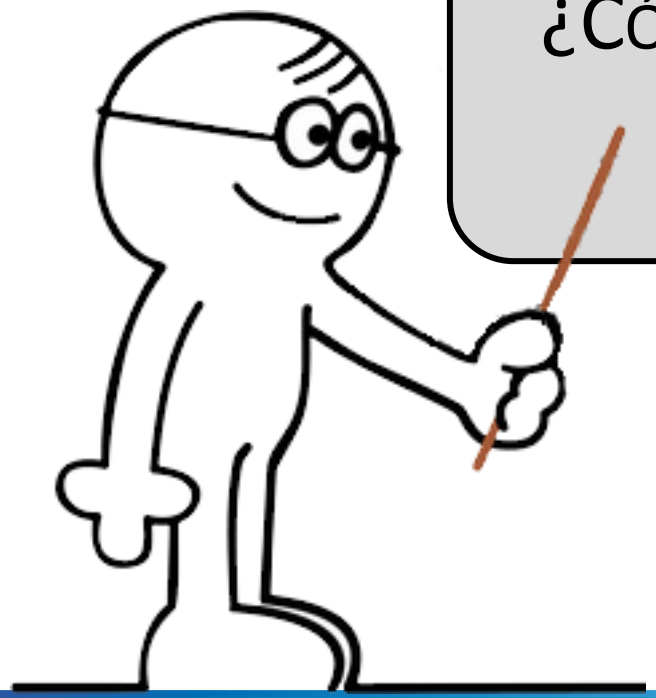
<tipo_dato> <nombre> Elementos del arreglo

```
Real      nota_final    =    {3.5, 4.0, 5.0, 2.8}
```

Imaginariamente, podemos pensar estos arreglos como:



¿Cómo se accede a los elementos de un Arreglo Unidimensional?



ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

- **ACCESO A UN ARREGLO:** para acceder a los elementos del arreglo se usan los corchetes [ ] y dentro de estos se especifica el índice de la posición a la que queremos acceder. Suponga que tenemos el siguiente arreglo:

```
                                0       1       2       3       4
Texto marcas = {"BMW", "Audi", "Mazda", "Ford", "KIA"}
```

PARA ACCEDER A UNA POSICIÓN ESPECÍFICA:

- Si queremos, por ejemplo, mostrar la marca en la casilla 1, deberíamos usar:

```
Escribir( marcas[1] )
```

- Ahora, para reemplazar la macar Ford por Toyota deberíamos hacerlos así:

```
marcas[3] = "Toyota"
```

PARA ACCEDER TODAS LAS POSICIONES DEL ARREGLO:

- Ahora, si lo que queremos es mostrar todos los datos del arreglo, entonces debemos recórrelo usando un ciclo, por ejemplo:

```
i=0
Mientras (i < 5) haga
  Escribir(marcas[i])
  i += 1
Fin_mientras
```

ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

- **ACCESO A UN ARREGLO:** los **arreglos**, como cualquier otra variable, deben ser declarados en la zona de declaración de variables del algoritmo. Recuerde que el operador paréntesis ( ) se usa para definir cuántos elementos tiene el arreglo, mientras que el operador corchete [ ] se usa para acceder a los elementos.

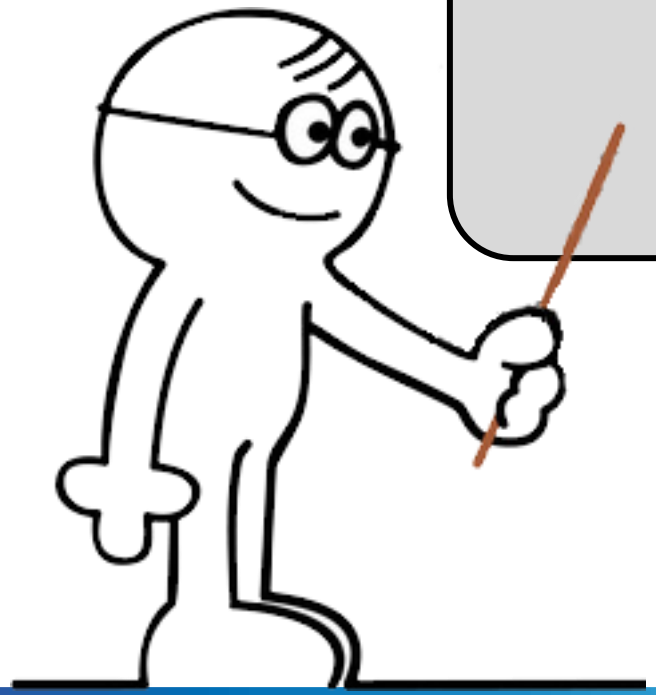
PSEUDOCÓDIGO

```
Real notas(4)
Texto nombres(4)

nombres[0] = “Oscar”
notas[0] = 3.5

Escribir(nombres[0], " obtuvo una nota de ", notas[0])
```

PYTHON

```
import numpy as np

notas = np.full((4), fill_value=0, dtype=float)
nombres = np.full((4), fill_value=None, dtype=object)

nombres[0] = "Oscar"
notas[0] = 3.5
print(f"{nombres[0]} obtuvo una nota de {notas[0]}")
```

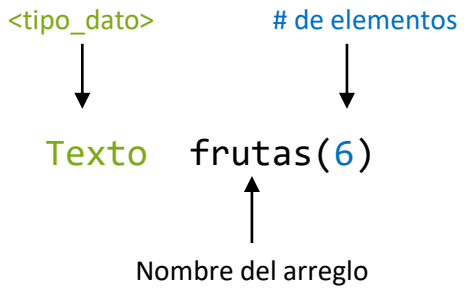
¿Cómo llenamos un Arreglo Unidimensional?



ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

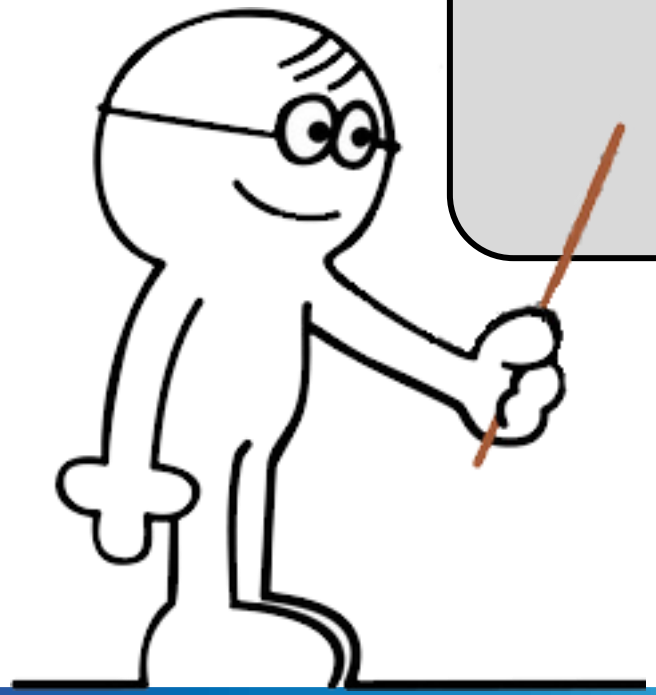
- **ACCESO A UN ARREGLO:** para llenar un arreglo debemos acceder a todas las casillas del arreglo. Nuevamente, para esto debemos usar un ciclo que nos permita movernos por todas las casillas del arreglo. Por ejemplo, definamos un arreglo para almacenar el nombre de 6 frutas.



Este arreglo está vacío, así que, si queremos llenarlo por completo, la forma más ágil de hacerlo es con ciclo de esta manera:

```
Para i=0 hasta 6, incremento 1 haga
   Escribir("Ingrese el nombre de la fruta # ", (i+1), ": ")
   Leer(frutas[i])
Fin_mientras
```

Ejemplos con Arreglos



ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

Haga un algoritmo que almacene 100 números enteros en un arreglo. Los números deben ser ingresados por el usuario.

ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

PSEUDOCÓDIGO

```
publico vacio leer_cien_numeros()

   Entero num(100), i

   Para i=0 hasta 100 incremento 1 haga
     Escribir("Ingrese un número entero: ")
     Leer(num[i])
   Fin_Para

Fin_Metodo
```

PYTHON

```
import numpy as np

def leer_cien_numeros():

   i = int
   num = np.full((100), fill_value=0, dtype=int)

   for i in range(100):
      num[i] = int(input("Ingrese un número entero: "))
```

ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

¿Qué tal si ahora intentamos hacer un algoritmo que pida 20 números y muestre los que son mayores al promedio de esos números?

ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

PSEUDOCÓDIGO

```
publico vacio numeros_mayores_al_promedio()
   Entero num(20), suma=0, i
   Real promedio

   # Pide los números y los acumula para el promedio
   Para i=0 hasta 20 incremento 1 haga
     Escribir("Ingrese un número entero: ")
     Leer(num[i])
     suma += num[i]
   Fin_Para

   # Calcula el promedio de los números ingresados
   promedio = suma/20
   # Muestra los números mayores al promedio
   Para i=0 hasta 20 incremento 1 haga
     Si (num[i] > promedio) entonces
        Escribir(num[i])
     Fin_Si
   Fin_Para
Fin_Metodo
```

PYTHON

```
import numpy as np

def numeros_mayores_al_promedio():

   suma = 0
   num = np.full((20), fill_value=0, dtype=int)

   # Pide los números y los acumula para el promedio
   for i in range(20):
      num[i] = int(input("Ingrese un número entero: "))
      suma += num[i]

   # Calcula el promedio de los números ingresados
   promedio = suma/20

   # Muestra los números mayores al promedio
   for i in range(20):
     if (num[i] > promedio):
       print(num[i])
```

ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

Ahora, diseñe un algoritmo que llene 2 arreglos, cada uno con 500 valores. El algoritmo debe restar los arreglos y mostrar los valores que son menores a cero, resultado de esta resta.

ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

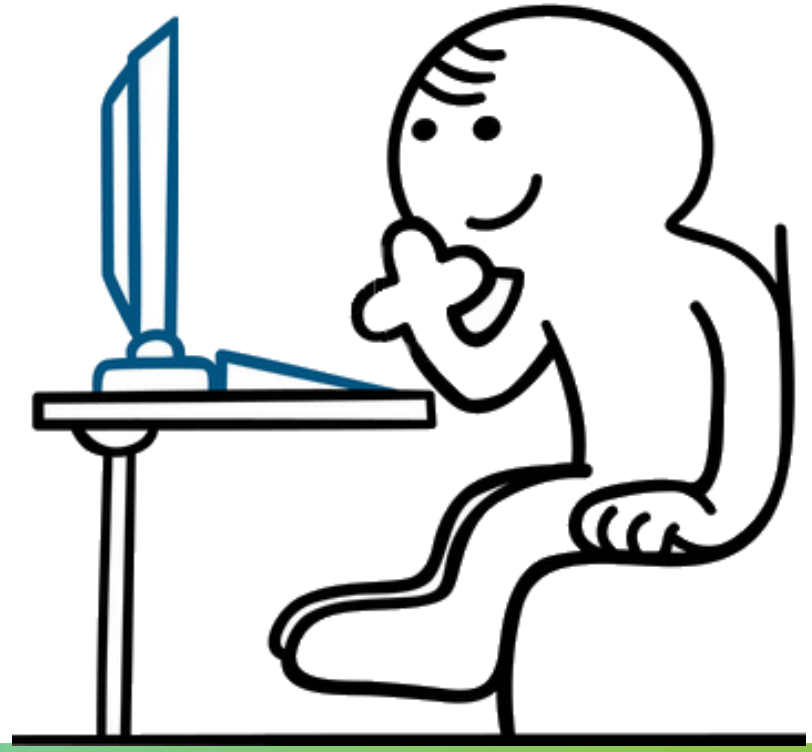
PSEUDOCÓDIGO

```
publico vacio restar_arreglos()
   Entero: i, cont=0, A(500), B(500), C(500)

   Para i=0 hasta 500 incremento 1 haga
     Escribir("Ingrese el valor en la posición", i+1, "del arreglo A:")
     Leer(A[i])
     Escribir("Ingrese el valor en la posición", i+1, "del arreglo B:")
     Leer(B[i])
     C[i] = A[i] - B[i]
     Si (C[i]  < 0) entonces
       cont = cont + 1
     Fin_si
   Fin_Para

   Si (cont == 0) entonces
    Escribir("No se generaron valores negativos en la resta")
   Sino
    Escribir("Los resultados negativos obtenidos de la resta son: ")
    Para i=0 hasta 500 incremento 1 haga
       Si (C[i] < 0) entonces
          Escribir(C[i])
       Fin_si
     Fin_Para
   Fin_si
 Fin_Metodo
```

ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

- **Ejemplo:** Haga un algoritmo que pida la información de los 1250 empleados de una empresa. El algoritmo debe almacenar el nombre, edad y el peso de cada empleado. El algoritmo debe mostrar la edad promedio entre todos los empleados y debe generar un listado con los empleados que pesen más de 100 kg y que sean menores a 25 años o mayores a 40 años.



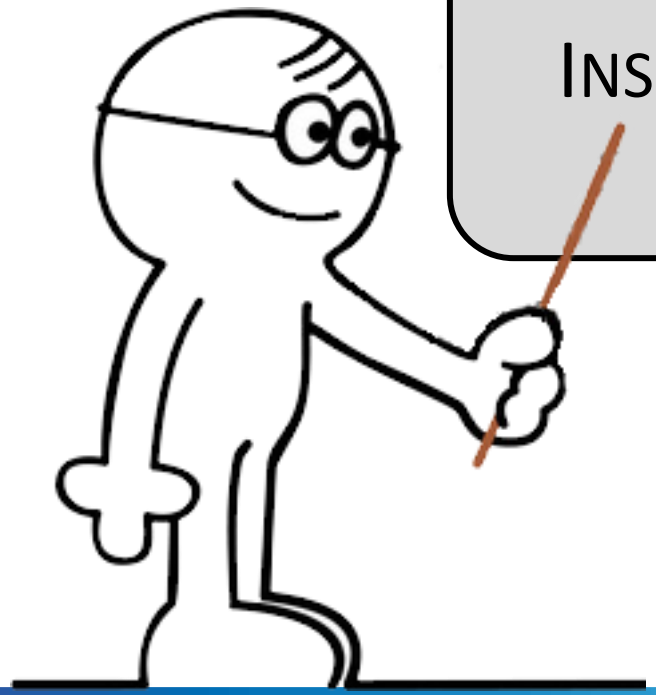
ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA
1803

PSEUDOCÓDIGO CON ARREGLOS PARALELOS

```
Clase EmpleadosEmpresa
 publico vacio registrar_empleados()
  Entero: i, edad(1250), sumaEdad=0, cont = 0
  Real: peso(1250)
  Cadena: nombre(1250)

   Para i=0 hasta 1250 incremento 1 haga
    Escribir("Ingrese el nombre del empleado # ", i+1, " : ")
    Leer(nombre[i])
    Escribir("Ingrese el peso y la edad del empleado ", nombre[i], " : ")
    Leer(peso[i], edad[i])
    sumaEdad = sumaEdad + edad
    Si (peso[i] > 100 and (edad[i] < 25 or edad[i] > 45)) entonces
      cont = cont+1
    Fin_si
   Fin_Para
   Escribir("La edad promedio de los empleados es: ", sumaEdad/1250)

   Si (cont == 0) entonces
    Escribir("No hay empleados que pesen más de 100 Kg y sean < de 25 o > de 45")
   Sino
    Escribir("Los empleados que pesan más de 100 Kg y que son < de 25 o > de 45 son: ")
    Para i=0 hasta 1250 incremento 1 haga
       Si (peso[i] > 100 and (edad[i] < 25 or edad[i] > 45)) entonces
          Escribir(nombre[i])
       Fin_si
     Fin_Para
   Fin_si
  Fin_metodo
Fin_Clase
```

Inserción de Elementos en un Arreglo



ARREGLOS UNIDIMENSIONALES

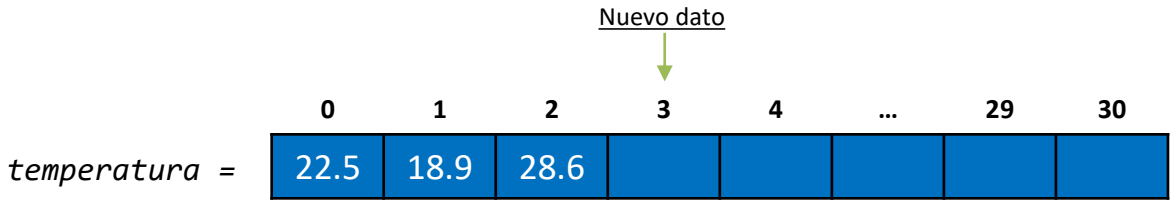
UNIVERSIDAD DE ANTIOQUIA

- **INSERCIÓN DE ELEMENTOS:**

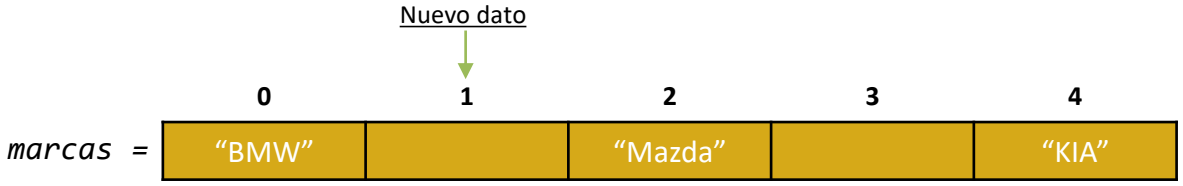
INSERCIÓN DE ELEMENTOS

La operación de inserción consiste en almacenar un dato en alguna casilla vacía del arreglo. Existen diferentes maneras de hacerlo, dos de ellas se explican a continuación.

Alternativa 1: Inserción después del último ingresado. En esta aproximación cada nuevo elemento se agrega después del último dato que se haya ingresado. Por ejemplo, en el siguiente arreglo el nuevo dato se almacenará en la casilla 3.



Alternativa 2: Se inserta el elemento en la primera casilla disponible. En esta aproximación el elemento se ingresa en la primera casilla que se encuentre que está disponible. Por ejemplo, en el siguiente arreglo el nuevo dato se almacenará en la casilla 1, que es la primera disponible.



ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

INSERCIÓN DE ELEMENTOS:

Alternativa 1

Para insertar el nuevo dato al final se usa un contador que determina cuántos datos almacenados, cuando se inserta un nuevo dato el contador aumenta.

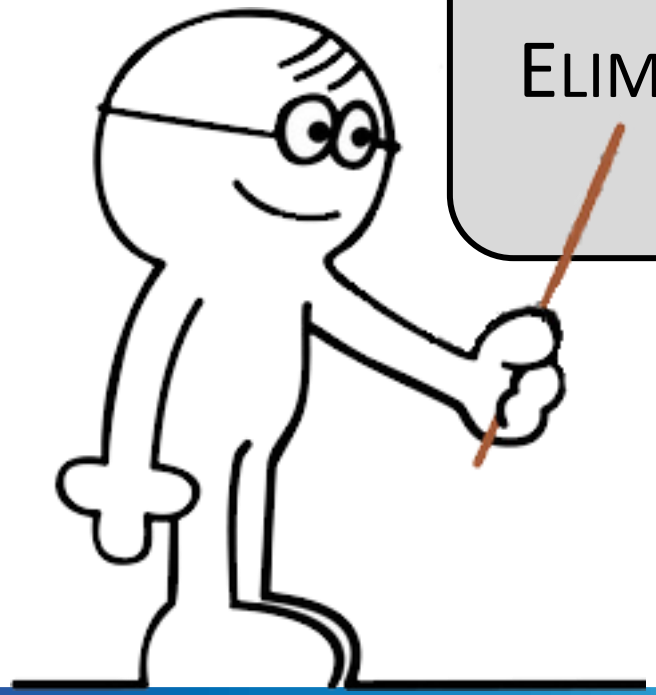
```
Entero cont_objetos = 0, n=10
Texto arreglo(n) = {None}

...

Si (cont_objetos  < len(arreglo)) Entonces
   arreglo[cont_objetos] = “Carlos”
   cont_objetos += 1
Fin_Si
```

Alternativa 2

En este caso se debe recorrer el arreglo y buscar una casilla “vacía”. Cuando se encuentra esa casilla se agrega el objeto y se aumenta el contador de objetos.

```
Entero cont_objetos = 0, n=10
Texto arreglo(n) = {None}
Si (cont_objetos  < len(arreglo)) Entonces
   Para i=0 hasta n, incremento 1 haga
      Si (arreglo[i] == None) Entonces
         arreglo[cont_objetos] = “Carlos”
         cont_objetos += 1
         romper
      Fin_Si
   Fin_Para
Fin_Si
```

Eliminación de Elementos en un Arreglo



ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

- **ELIMINACIÓN DE ELEMENTOS:**

La eliminación de datos en un arreglo consiste en asignar un dato “vacío” a la casilla donde está el dato que queremos eliminar. Igual que antes, hay dos alternativas.

Alternativa 1: mover los elementos hacia la izquierda después de eliminar uno. En esta aproximación después de eliminar un elemento, todos los demás, delante del recién borrado deben moverse a la izquierda.

Alternativa 2: asignar un valor “vacío a la casilla recién eliminada”. En esta aproximación después de eliminar un elemento, todos los demás, delante del recién borrado deben moverse a la izquierda.

ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

ELIMINACIÓN DE ELEMENTOS:

Alternativa 1

En este caso se deben mover todos los datos hacia la izquierda, después de encontrar el dato que se va a borrar

```
Entero cont_objetos = 0, i=0, n=5
Texto arreglo (n) = {“A”, “B”, “C”, “D”, “E”}
Texto a_borrar = “C”

Mientras (i < cont_objetos) Entonces
   Si (arreglo[i] == a_borrar)Entonces
      Para j=1 hasta n-1, incremento 1 haga
         arreglo[j] = arreglo[j+1]
      Fin_Para
      cont_objetos -= 1
   Fin_Si
Fin_Mientras
```

Alternativa 2

En este caso cuando se encuentra el dato, la casilla se marca como “vacía”.

```
Entero cont_objetos = 0, i=0, n=5
Texto arreglo (n) = {“A”, “B”, “C”, “D”, “E”}
Texto a_borrar = “C”

Mientras (i < n) Entonces
   Si (arreglo[i] == a_borrar)Entonces
      arreglo[i] = None
      cont_objetos -= 1
   Fin_Si
Fin_Mientras
```

LÓGICA Y REPRESENTACIÓN I

PROGRAMA DE INGENIERÍA DE SISTEMAS

Material desarrollado por **Carlos Andrés Mera Banguero**
https://github.com/carlosmera20/