LÓGICA Y REPRESENTACIÓN I

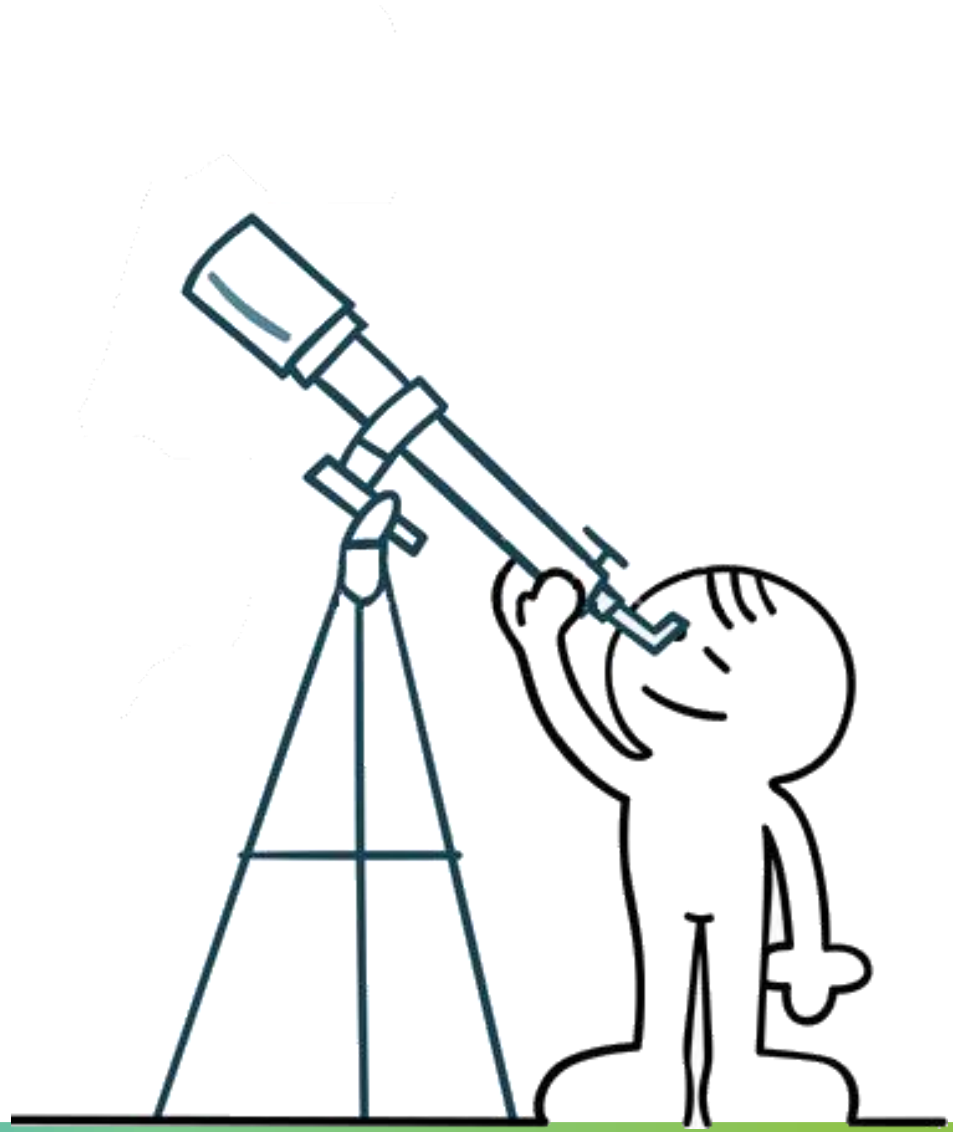
PROGRAMA DE INGENIERÍA DE SISTEMAS

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

CONTENIDO

UNIVERSIDAD DE ANTIOQUIA

- **ESTRUCTURAS DE ALMACENAMIENTO ESTÁTICAS - ARREGLOS**
  - Ordenamiento de arreglos: burbuja, selección, e inserción
  - Búsqueda lineal y búsqueda binaria



Ordenamiento de Arreglos

ORDENAMIENTO DE ARREGLOS

UNIVERSIDAD DE ANTIOQUIA

ORDENAMIENTO

El ordenamiento de un arreglo consiste en reorganizar los elementos del arreglo en un orden específico, bien sea de manera ascendente o descendente. Por ejemplo, dado el arreglo:

Índice →	0	1	2	3
Elementos →	2	8	9	6

La versión ordena de mayor a menor de este arreglo es:

Índice →	0	1	2	3
Elementos →	9	8	6	2

EXISTEN TRES MÉTODOS BÁSICOS PARA ORDENAR UN ARREGLO:

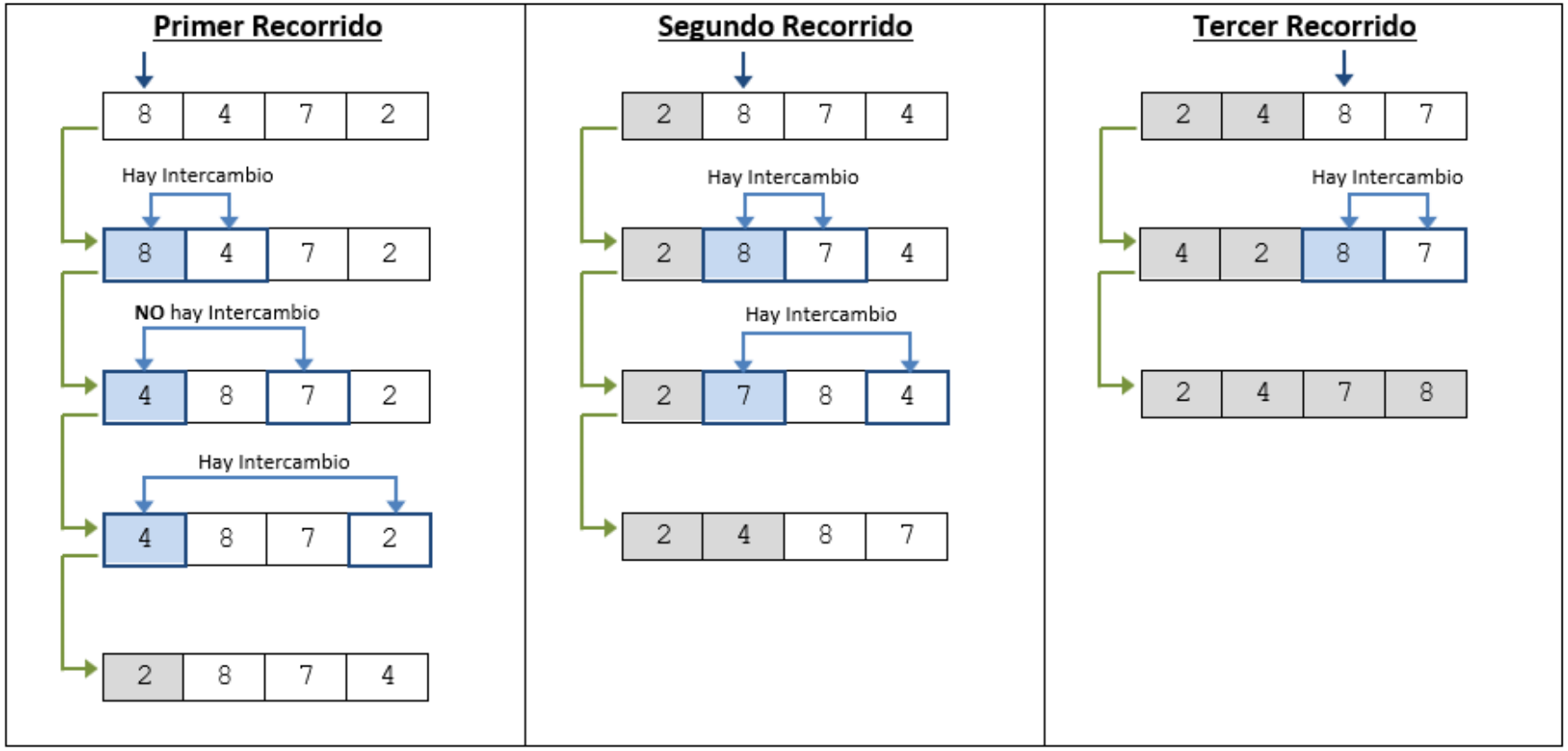
- **Método de la Burbuja**. Es uno de los algoritmos más simples, pero también uno de los más lentos y consiste en que múltiples veces se comparan las parejas de elementos adyacentes del arreglo, **burbujeando** los elementos más grandes al final del arreglo.
- **Método de Selección**, el cual se centra en buscar, en cada iteración, el elemento más pequeño o más grande de los elementos que no se han ordenado.
- **Método de Inserción**, tiende a ser eficiente para arreglos pequeños casi ordenados y su funcionamiento consiste en comparar cada elemento del arreglo que no está ordenado, con los elementos ya ordenados. Cuando se encuentra la posición correcta del elemento que se está ordenando se inserta en donde debe ir y se desplazan los demás elementos a la derecha.

Métodos de Ordenamiento

ORDENAMIENTO DE ARREGLOS

UNIVERSIDAD DE ANTIOQUIA

- **MÉTODO DE SELECCIÓN:** una de las estrategias más comunes para ordenar un arreglo es utilizar el método de selección el cual consiste en buscar el elemento menor y ubicarlo en la primera casilla del arreglo; luego, se busca el segundo elemento menor y se ubica en la segunda casilla y, así, sucesivamente hasta lograr que todos los elementos queden ordenados.



ORDENAMIENTO DE ARREGLOS

UNIVERSIDAD DE ANTIOQUIA

- **MÉTODO DE SELECCIÓN:** veamos el algoritmo de ordenamiento de selección implementado en una función que recibe el arreglo y retorna una copia del arreglo ordenada de menor a mayor.

PSEUDOCÓDIGO

```
publico Real() ordenar_por_seleccion(Real() arr)
  Real arr2 = arr.copiar(), aux
  Entero i, j, n = arr.length()

  Para i=0 hasta n-1 incremento 1 Haga
    Para j=i+1 hasta n incremento 1 Haga
      # Se verifica si el menor está a la derecha para moverlo
      Si (arr2[j] < arr2[i]) Entonces
        aux = arr2[i]
        arr2[i] = arr2[j]
        arr2[j] = aux
      Fin_Si
    Fin_Para
  Fin_Para
  Retornar arr2
Fin_Metodo
```

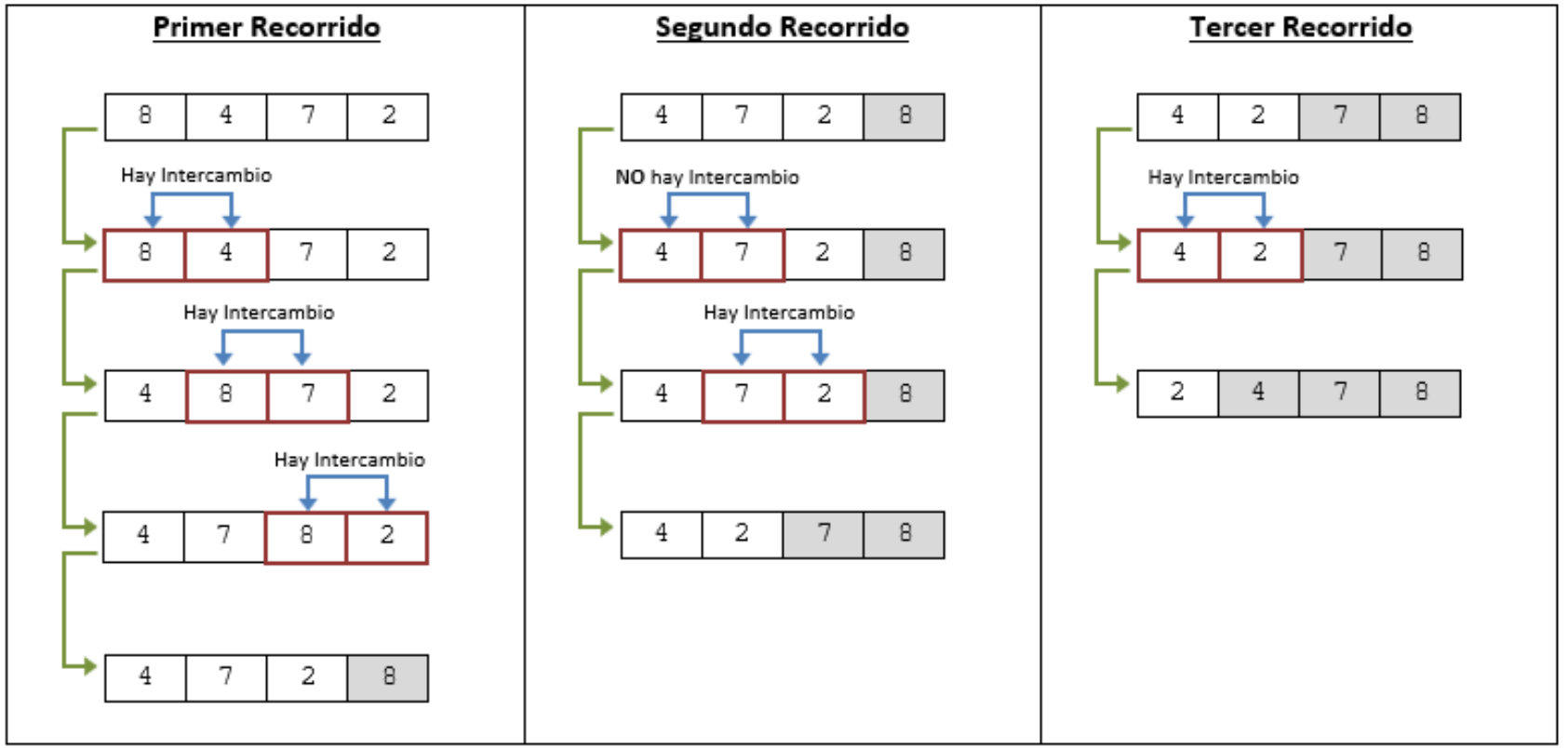
PYTHON

```
def ordenar_por_ seleccion(arr):
  arr2 = arr.copy()
  n = arr.length()

  for i in range(n-1):
    for j in range (n):
      # Se verifica si el menor está a la derecha para moverlo
      if (arr2[j] < arr2[i]):
        aux = arr2[i]
        arr2[i] = arr2[j]
        arr2[j] = aux
  return arr2
```

ORDENAMIENTO DE ARREGLOS

UNIVERSIDAD DE ANTIOQUIA

- **MÉTODO DE LA BURBUJA:** es un método simple en el que se comparan las parejas de elementos adyacentes del arreglo, intercambiándolos para que el más grande se **burbujee** hasta el final del arreglo. Esta operación se debe hacer múltiples veces, haciendo tantos recorridos del arreglo, como elementos tenga este.



ORDENAMIENTO DE ARREGLOS

UNIVERSIDAD DE ANTIOQUIA

- **MÉTODO DE LA BURBUJA:** veamos el algoritmo de ordenamiento de la burbuja implementado en una función que recibe el arreglo y retorna una copia del arreglo ordenada de menor a mayor.

PSEUDOCÓDIGO

```
publico Real() ordenar_por_burbuja(Real() arr)
  Real arr2 = arr.copiar(), aux
  Entero i, j, n = arr.length()

  Para i=0 hasta n-1 incremento 1 Haga
    Para j=0 hasta n-1-i incremento 1 Haga
      # Se verifica si el mayor está a la izquierda
      Si (arr2[j] > arr2[j+1]) Entonces
        aux = arr2[j]
        arr2[j] = arr2[j+1]
        arr2[j+1] = aux
      Fin_Si
    Fin_Para
  Fin_Para
  Retornar arr2
Fin_Metodo
```

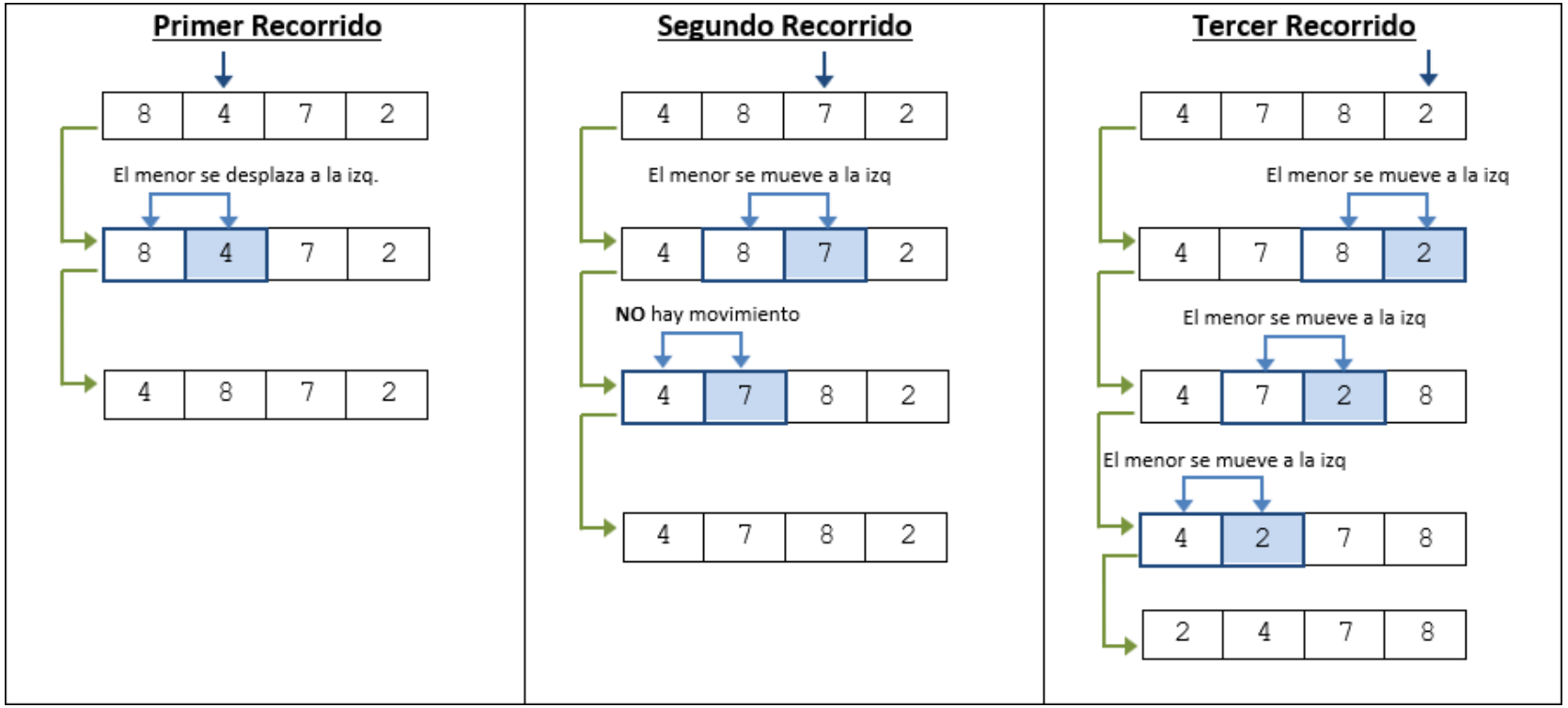
PYTHON

```
def ordenar_por_burbuja(arr):
  arr2 = arr.copy()
  n = arr.length()

  for i in range(n-1):
    for j in range (n-1-i):
      # Se verifica si el mayor está a la izquierda
      if (arr2[j] > arr2[j+1]):
        aux = arr2[j]
        arr2[j] = arr2[j+1]
        arr2[j+1] = aux
  return arr2
```

ORDENAMIENTO DE ARREGLOS

UNIVERSIDAD DE ANTIOQUIA

- **MÉTODO DE INSERCIÓN:** este método ordena de manera progresiva el arreglo, recorriéndolo y buscando en qué posición debe ir cada elemento para insertarlo y desplazar los elementos siguientes hacia derecha.



ORDENAMIENTO DE ARREGLOS

UNIVERSIDAD DE ANTIOQUIA

- **MÉTODO DE INSERCIÓN:** veamos el algoritmo de ordenamiento de selección implementado en una función que recibe el arreglo y retorna una copia del arreglo ordenada de menor a mayor.

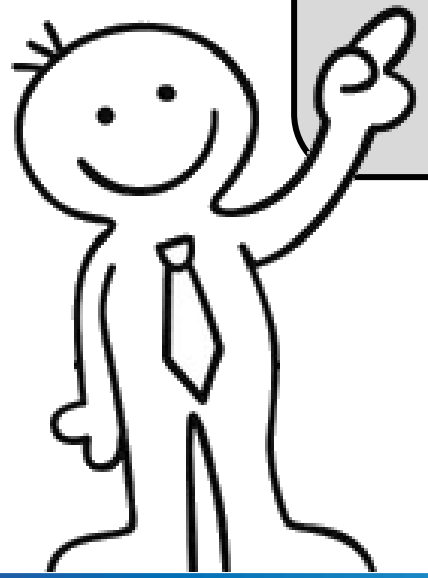
PSEUDOCÓDIGO

```
publico Real() ordenar_por_inserción(Real() arr)
  Real arr2 = arr.copiar(), aux
  Entero i, j, n = arr.length()

  Para i=1 hasta n incremento 1 Haga
    aux = arr2[i]
    j = i-1
    Mientras (j >= 0 and arr2[j] > aux) Haga
      arr2[j+1] = arr2[j]
      j = j-1
    Fin_Mientras
    arr2[j+1] = aux
  Fin_Para
  Retornar arr2
Fin_Metodo
```

PYTHON

```
def ordenar_por_ seleccion(arr):
  arr2 = arr.copy()
  n = arr.length()

  for i in range(1, n):
    aux = arr2[i]
    j = i-1
    while(j >= 0 and arr2[j] > aux):
      arr2[j+1] = arr2[j]
      j = j-1
    arr2[j+1] = aux
  return arr2
```

Algunos ejemplos que requieren ordenamiento de arreglos



ORDENAMIENTO DE ARREGLOS

UNIVERSIDAD DE ANTIOQUIA
1803

EJERCICIO: Haga un algoritmo que lea los nombre y las notas finales de los 35 estudiantes del curso de Fundamentos de Programación. El algoritmo debe mostrar el listado de estudiantes y sus notas, ordenados de la mayor nota a la menor.

ORDENAMIENTO DE ARREGLOS

UNIVERSIDAD DE ANTIOQUIA

- **Ejemplo:** Haga un algoritmo que lea los nombre y las notas finales de los 35 estudiantes del curso de Fundamentos de Programación. El algoritmo debe mostrar el listado de estudiantes y sus notas, ordenados de la mayor nota a la menor.

```
Clase NotasOrdenasDeFundamentos
 Metodo principal()
   Entero i, j
   Real notas(35), auxNota
   Cadena nombres(35), auxNombre

   //Se piden los nombres y notas de los estudiantes
   Para i=0 hasta len(notas) haga
    Escribir("Ingrese el nombre del estudiante:")
    Leer(nombres[i])
    Escribir("Ingrese la nota del estudiante:")
    Leer(notas[i])
   Fin_Para

   //Se ordenan los dos arreglos usando las notas
   Para i=0 hasta len(notas) haga
     Para j=i+1 hasta j < len(notas) haga
      Si (notas[j] > notas[i]) entonces
        auxNota = notas[i]
        notas[i] = notas[j]
        notas[j] = auxNota
        //Los cambios que se hacen en el
        //arreglo de las notas se deben hacer
        //en el arreglo de los nombres
        auxNombre = nombres[i]
        nombres[i] = nombres[j]
        nombres[j] = auxNombre
      Fin_Si
     Fin_Para
   Fin_Para

   //Se muestra el listado ordenado por nota
   Escribir("Listado ordenado de notas:")
   Para i=0 hasta len(notas) haga
     Escribir(nombres[i], "     ", notas[i])
   Fin_Para
  Fin_Metodo
 Fin_Clase
```

ORDENAMIENTO DE ARREGLOS

UNIVERSIDAD DE ANTIOQUIA

EJERCICIO: 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 generar un listado con los empleados que pesen más de 100 kg y que sean menores a 25 años o mayores a 40 años. Dicho listado debe estar ordenado por la edad de los empleados de mayor a menor.

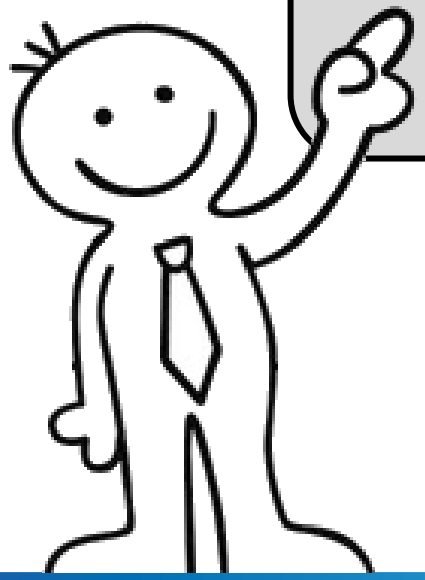
ORDENAMIENTO DE ARREGLOS

UNIVERSIDAD DE ANTIOQUIA

PSEUDOCÓDIGO

```
clase EmpleadosEmpresa
 publico vacio procesar_empleados()
  Entero N=120, i, j, edad(N), aux1
  Real peso(N), aux2
  Cadena nombre(N), aux3, listado=“”

  Para i=0 hasta N haga
    Escribir(“Ingrese el nombre, edad y peso del empleado # ”, i+1)
    Leer(nombre[i], edad[i], peso[i])
  Fin_Para

   Para i=0 hasta N haga
     Para j=i+1 hasta N haga
       Si (edad[j] > edad[i]) entonces
        aux1 = edad[i]
        edad[i] = edad[j]
        edad[j] = aux

        aux2 = peso[i]
        peso[i] = peso[j]
        peso[j] = aux2

        aux3 = nombre[i]
        nombre[i] = nombre[j]
        nombre[j] = aux3
      Fin_Si
    Fin_Para
  Fin_Para

  #Como primero se ordenó el arreglo, entonces el listado sale ordenado por la edad
  Escribir(“Los empleados que pesan más de 100 Kg y que son < de 25 o > de 45 son: ”)

  Para i=0 hasta N haga
    Si (peso[i] > 100 and (edad[i] < 25 or edad[i] > 45)) entonces
      Escribir(nombre[i], “\t”, edad[i] , “\t”, peso[i] , “\n”)
    Fin_si
  Fin_Para
 Fin_metodo
Fin_Clase
```

Métodos de Búsqueda de Datos en un Arreglo



ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA
1803

- **BÚSQUEDA LINEAL:** la estrategia más común para buscar un valor en un arreglo es el método de búsqueda lineal el cual consiste en recorrer el arreglo secuencialmente comparando el valor de cada elemento contra el valor buscado. Por ejemplo, dado el siguiente arreglo, haga un algoritmo que diga en qué posición se encuentra el valor 3.5:

X=	5.0	4.0	3.5	2.87

ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

- **BÚSQUEDA LINEAL:** la estrategia más común para buscar un valor en un arreglo es el método de búsqueda lineal el cual consiste en recorrer el arreglo secuencialmente comparando el valor de cada elemento contra el valor buscado. Por ejemplo, dado el siguiente arreglo, haga un algoritmo que diga en qué posición se encuentra el valor 3.5:

X=	5.0	4.0	3.5	2.87

```
publico vacio busqueda_lineal(Real() arr, Real valorBuscado)
   Entero i, pos=-1

   Para i=0 hasta len(arreglo) haga
     Si (arr[i] == valorBuscado) entonces
       pos = i
       Escribir("El valor se encontró en la posición ", i)
     Fin_Si
   Fin_Para
   Si (pos == -1) Entonces
     Escribir("El valor NO se encontró")
   Fin_Si
 Fin_Metodo
```

ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA
1803

- **BÚSQUEDA BINARIA:** la búsqueda binaria es un método de búsqueda que se ejecuta sobre un arreglo previamente ordenado y funciona dividiendo repetidamente el espacio de búsqueda a la mitad hasta que se encuentra el elemento deseado o se determina que el elemento no está presente en el arreglo.Por ejemplo, dado el siguiente arreglo, haga un algoritmo que diga en qué posición se encuentra el valor 3.5:

X=	5.0	4.0	3.5	2.87

ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

- **BÚSQUEDA BINARIA:**

X =	5.0	4.0	3.5	2.87

```
publico vacio busqueda_binaria(Real() arr, Real valorBuscado)
  Entero inicio = 0, fin = len(arr)-1 , medio, pos=-1
  Real elemento_medio
  Mientras (inicio <= fin) Haga
    medio = (inicio + fin) // 2
    elemento_medio = arr[medio]
    Si (elemento_medio == valorBuscado) Entonces
      pos = medio
      Escribir("El valor se encontró en la posición ", pos)
    Sino
      Si (elemento_medio < valorBuscado) Entonces
        inicio = medio + 1
      Sino
        fin = medio - 1
      Fin_Si
    Fin_Si
  Fin_Mientras
  Si (pos == -1) Entonces
    Escribir("El valor NO se encontró")
  Fin_Si
Fin_Metodo
```

Un par de Ejercicios

ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

EJERCICIO: Haga un algoritmo que solicite N números enteros al usuario y los almacene en un arreglo. El valor de N debe solicitarse al usuario. Una vez se ha llenado el arreglo, el algoritmo debe invertir el orden de sus elementos. Para finalizar el algoritmo debe mostrar el cuadrado de los elementos (invertidos).

ARREGLOS UNIDIMENSIONALES

UNIVERSIDAD DE ANTIOQUIA

EJERCICIO: La EPS "***No te mueras aquí, por favor***" está haciendo una campaña de sensibilización sobre los problemas que puede traer para la salud el exceso de azúcar en la sangre.

¡Como parte de esta campaña, la EPS ha decidido realizar hasta 1000 exámenes de glucosa gratis!. Para cada muestra la EPS necesita que usted desarrolle un algoritmo que almacene el nombre de la persona, la cédula, la edad, el sexo y el nivel de azúcar en la sangre (o glicemia) de cada persona que se hace el examen.

El programa debe tener un menú que permita:

- Registrar un examen de glucosa
- Consultar la información de uno de los pacientes por cédula
- Conocer cuál o cuáles son las personas con el nivel de azúcar más alto
- Generar el listado (nombre, edad y nivel de azúcar en la sangre) de aquellas mujeres de más de 40 años cuyo nivel de azúcar está por encima de 126 mg/dl, ordenado de mayor a menor nivel de azúcar.

LÓGICA Y REPRESENTACIÓN I

PROGRAMA DE INGENIERÍA DE SISTEMAS

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