



# PROGRAMACIÓN I

# ARREGLOS



# Operaciones con Arreglos

- Cargar un arreglo.
- Recorrer un arreglo.
- Buscar un elemento en particular.
- Acceder a un elemento en una posición determinada.
- Acceder a una posición determinada y mostrar su contenido.
- Insertar un nuevo elemento.
- Eliminar un elemento.
- Ordenar un arreglo

# **Operaciones con Arreglos**

- **Cargar un arreglo.**
- **Recorrer un arreglo para mostrar sus elementos.**

# Búsqueda

Un algoritmo de búsqueda es aquel que está diseñado para localizar un elemento concreto dentro de un arreglo.

## Tipos de Búsqueda:

- Arreglo desordenado sin elementos repetidos.
- Arreglo desordenado con elementos repetidos.
- Arreglo ordenado sin elementos repetidos: BÚSQUEDA BINARIA
- Arreglo ordenado con elementos repetidos.



**Acción Cargar\_Arreglo es**

**Ambiente**

**lista[10]:entero;  
i,b,enc,a:entero;**

**Algoritmo**

**para (i = 0; i < 10; i:=i+1)  
    escribir("Ingresar el elemento");  
    leer(a);  
    lista[i]:=a;**

**fin para**

**para (i = 0; i < 10; i:=i+1)  
    Escribir (lista[i]);  
    si (lista[i]%2=0 escribir ("Es par"));  
    fin si  
fin para**

**Escribir ("Ingrese elemento a buscar");**

**Leer (b);**

**i:=0;**

**mientras (i<10  $\wedge$  enc=0)  
    si (b= lista[i]) enc:=1;  
        sino i:=i+1;  
    fin si**

**Fin mientras**

**si (enc=1) escribir ("Número encontrado ");  
    sino escribir("Número no encontrado ");**

**fin si**

**Fin acción**

**Búsqueda en un arreglo sin  
elementos repetidos**

**En el caso del programa anterior, los elementos del arreglo están desordenado.**

**Pregunta: Si los elementos del arreglo estarían ordenados, ¿funcionaría el programa de búsqueda?**



**Acción** Cargar\_Arreglo es

**Ambiente**

lista[10]:entero;  
i,b,enc,a:entero;

**Algoritmo**

```
para (i = 0; i < 10; i:=i+1)
    escribir("Ingresar el elemento");
    leer(a);
    lista[i]:=a;
fin para

para (i = 0; i < 10; i:=i+1)
    Escribir (lista[i]);
    si (lista[i]%2=0 escribir ("Es par"));
    fin si
fin ipara

Escribir ("Ingrese elemento a buscar");
Leer (b);
i:=0;
mientras (i<10 )
    si (b= lista[i]) enc:=enc+1
    fin si
    i:=i+1;
fin mientras

si (enc=0) escribir ("Elemento no encontrado ");
sino escribir("Elemento encontrado: ",enc);
fin si
Fin acción
```

**¿Qué hace este programa?**



## ACTIVIDAD

**Modificar el programa anterior para que funcione en un arreglo ordenado con elementos repetidos como por ejemplo:**

**$A=\{1,2,3,3,5,6,7,7,8\}$**



**Acción Cargar\_Arreglo es**

**Ambiente**

lista[10]:entero;  
i,b,enc,a:entero;

**Algoritmo**

```
para (i = 0; i < 10; i:=i+1)
    escribir("Ingresar el elemento");
    leer(a);
    lista[i]:=a;
fin para

para (i = 0; i < 10; i:=i+1)
    Escribir (lista[i]);
    si (lista[i]%2=0 escribir ("Es par"));
    fin si
fin ipara

Escribir ("Ingrese elemento a buscar");
Leer (b);
i:=0; enc:=0;
mientras ( i <10)
    mientras (b= lista[i]) hacer
        enc:=enc+1
        i:=i+1;
    fin mientras
i:=10;
fin mientras

si (enc=0) escribir ("Elemento no encontrado ");
    sino escribir("Elemento encontrado: ",enc);
fin si
Fin acción
```

**Arreglo ordenado con  
elementos repetidos**

# Búsqueda Binaria

La **búsqueda binaria** o **búsqueda dicotómica** es un algoritmo de búsqueda.

Para realizarla, es necesario contar con un arreglo ordenado.

Se toma un elemento central, normalmente el elemento que se encuentra a la mitad del arreglo, y se lo compara con el elemento buscado. Si el elemento buscado es menor, se toma el intervalo que va desde el elemento central al principio, en caso contrario, se toma el intervalo que va desde el elemento central hasta el final del intervalo.

Se procede de esta manera con intervalos cada vez menores hasta que se llega a un intervalo indivisible, en cuyo caso el elemento no está en el vector, o el elemento central sea el elemento buscado.

0	1	2	3	4	5	6	7	8
-8	4	5	9	12	18	25	40	60

$$izq=0$$

dch=8

**centro=4**

$$40 > 12$$

5	6	7	8
18	25	40	60

$$izq=5$$

dch=8

**centro=6**

$$40 > 25$$

7	8
40	60

$$izq=7$$

dch=8

**centro=7**

Encontrado

## ¿es correcto este trozo de algoritmo?

**Primero := 0;**

**Ultimo := tam – 1**

**Encontrado := falso**

**mientras ( Primero < Ultimo )  $\wedge$  ( Encontrado = falso )**

**Central = ( Primero + Ultimo ) / 2**

**si ( Clave = A [ Central ] ) entonces Encontrado = verdadero**

**sino**

**si ( Clave > A [ central ] ) entonces Primero = Central + 1**

**Sino Ultimo = Central – 1**

**fin si**

**fin si**

**fin mientras**

**si ( Encontrado == verdadero ) entonces escribir (“Elemento encontrado”);**

**sino escribir ( “No se ha encontrado”);**

**fin si**

# Operación de Inserción

- 1) Verificar que el arreglo tenga posiciones disponibles para insertar un nuevo elemento.
- 2) Si existe posiciones disponibles proceder a realizar la inserción.
- 3) Si el arreglo esta ordenado se debe realizar la inserción en la posición que corresponda para que quede ordenado después de realizar la misma (al inicio, al medio o al final).
- 4) Si el arreglo no esta ordenado se puede optar por insertar el nuevo elemento al final del arreglo.
- 5) Para realizar la inserción , se debe mover los elementos una posición hacia la derecha. Es necesario conocer la posición del último elemento del arreglo.



Tipos de Inserción que se pueden realizar en arreglos con las siguientes condiciones:

# Operación de Inserción

- Arreglo ordenado sin elementos repetidos
- Arreglo ordenado con elementos repetidos.
- Arreglo desordenado sin elementos repetidos.
- Arreglo desordenado con elementos repetidos.

# Operación de Inserción

Arreglo Original, desordenado

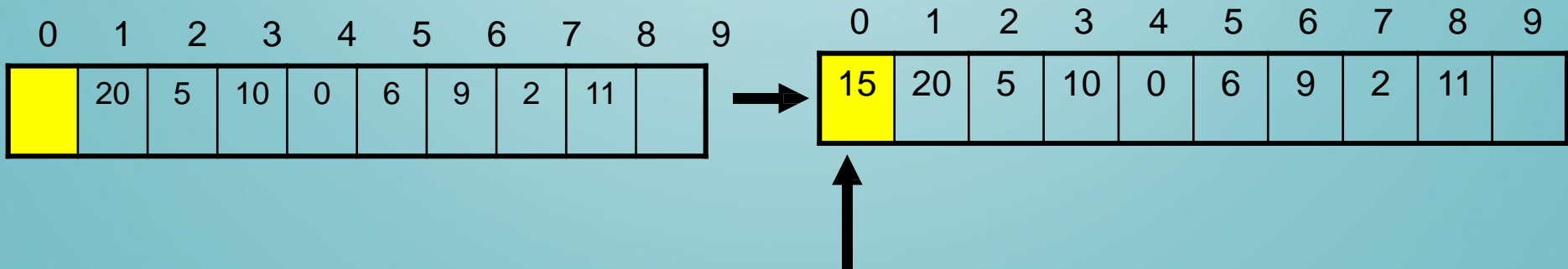
0	1	2	3	4	5	6	7	8	9
20	5	10	0	6	9	2	11		

Tiene libre las posiciones 8 y 9.

Tiene libre las posiciones 8 y 9. Por Ejemplo Insertar el elemento 15 en la posición 8. El nuevo arreglo después de la inserción es:

20	5	10	0	6	9	2	11	15	
----	---	----	---	---	---	---	----	----	--

También se puede optar por insertar en la posición 1. En este caso se deben mover los elementos una posición hacia la derecha, dejar libre la posición 1 e insertar el nuevo elemento. Por Ejemplo Insertar el elemento 15 en la posición 0. El nuevo arreglo después de la inserción es:



# Operación de Inserción

Arreglo Original , ordenado sin elemento repetidos

0	1	2	3	4	5	6	7	8	9
5	7	10	12	13	16	18	20		

Tiene libre las posiciones 8 y 9.

Tiene libre las posiciones 8 y 9, por lo tanto hay lugares disponibles para insertar.

El elemento que se ingresa se debe colocar en la posición que corresponda para que el arreglo continúe ordenado después de realizar la inserción del nuevo elemento.

Una vez que se ingresa el elemento a insertar, se debe preguntar si la inserción es:

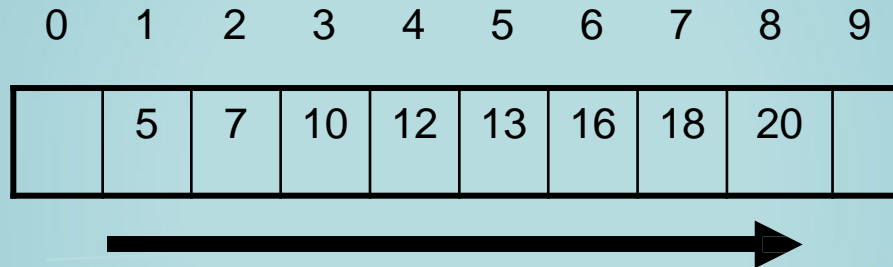
- Al inicio ,antes del primer elemento
- Al final del arreglo
- Al medio del arreglo

## Insertar al inicio del arreglo

Por ejemplo insertar el número 3.

$3 < 5$ , entonces puedo realizar la inserción.

Desplazo los elementos del arreglo una posición a la derecha, quedando la posición 0 disponible para insertar 3.



```
/*Desplazar los elementos del arreglo una posición hacia la derecha */  
for (j=i; j<ult;j++)  
    a[j+1]= a[j];
```

Insertar el número 3\_



## Insertar al final del arreglo

Por ejemplo insertar el número 22.

$20 < 22$ , entonces puedo realizar la inserción al final del arreglo.


0	1	2	3	4	5	6	7	8	9
5	7	10	12	13	16	18	20	22	

## Insertar al medio del arreglo

Por ejemplo insertar el número 11.

Tengo que recorrer el arreglo hasta encontrar el primer número mayor, en este caso 12.  
A partir de 12 tengo que desplazar los elementos una posición hacia la derecha, para dejar un lugar libre en la posición 3

0	1	2	3	4	5	6	7	8	9
5	7	10		12	13	16	18	20	



/\*Desplazar los elementos del arreglo una posición hacia la derecha \*/  
for (j=ult; j>i;j--)  
    a[j+1]= a[j];

Ahora se puede insertar el número 11 en la posición 3.

0	1	2	3	4	5	6	7	8	9
5	7	10	11	12	13	16	18	20	



# Operación de Eliminación

- 1) Verificar que el arreglo se encuentre el elemento a eliminar.
- 2) Si existe el elemento en el arreglo proceder a realizar la eliminación.
- 3) La eliminación se realiza moviendo los elementos una posición hacia la izquierda los elementos del arreglo, quedando una posición libre. Para esto es necesario identificar la posición en la que se encuentra el elemento a eliminar .
- 4) Luego de realizar la eliminación el arreglo cuenta con un elemento menos.

## Arreglo Original

0	1	2	3	4	5	6	7	8	9
20	5	10	0	6	9	2	11	7	3

Eliminar el elemento 9.

Al eliminar el elemento 9 en la posición 5. El nuevo arreglo después de la eliminación es:

20	5	10	0	6	2	11	7	3	
----	---	----	---	---	---	----	---	---	--

Al eliminar el elemento 9, queda libre la última posición

Tipos de Eliminación se pueden realizar en arreglos con las siguientes condiciones:

## Operación de Eliminación

- Arreglo ordenado sin elementos repetidos. En este caso se puede utilizar la búsqueda binaria para localizar el elemento a eliminar.
- Arreglo ordenado con elementos repetidos.
- Arreglo desordenado sin elementos repetidos.
- Arreglo desordenado con elementos repetidos.

# Operación de eliminación en un arreglo desordenado sin elementos repetidos

Acción Eliminar es  
Ambiente

.....  
.....

algoritmo

.....  
.....

/\*Buscar elemento en el arreglo

i=0;

mientras (n!=a[i] && i<= max)

i++;

Fin mientras

si (a[i]== n)

/\*Eliminar el elemento n del arreglo \*/

para (j=i; j<max;j++)

a[j]= a[j+1];

/\*Mostrar el arreglo despues de la eliminación\*/

Para (i = 0; i < max-1; i++)

Escribir ("Digito",a[i]);

Fin para

Sino escribir("Elemento no está en el arreglo\n");

fin si

Fin acción