# Algoritmos y Estructuras de Datos

## Clase de Práctica 13 - Arreglos unidimensionales

Comisión A y F

Docentes de práctica:

**Auxiliar Tomás Assenza**

**Auxiliar Facundo Sanchez**

Universidad Tecnológica Nacional - Facultad Regional Santa Fe

# Búsqueda lineal

- ¿Cuando la utilizamos?

# Búsqueda lineal

- ¿Cuando la utilizamos?

# Búsqueda binaria

- ¿Cuándo la utilizamos?

# Búsqueda lineal

```
int buscaElemento(int arreglo[], int tam, int elemento) {
```

# Búsqueda lineal

```
int buscaElemento(int arreglo[], int tam, int elemento) {
  int i = -1;
  bool encontrado = false;
```

# Búsqueda lineal

```
int buscaElemento(int arreglo[], int tam, int elemento) {
  int i = -1;
  bool encontrado = false;

  while (!encontrado and i < tam) {
    i++;
    if (arreglo[i] == elemento) encontrado = true;
  }
```

# Búsqueda lineal

```
int buscaElemento(int arreglo[], int tam, int elemento) {
  int i = -1;
  bool encontrado = false;

  while (!encontrado and i < tam) {
    i++;
    if (arreglo[i] == elemento) encontrado = true;
  }
  if (i == tam) i = -1;
  return i;
}
```

# Búsqueda binaria (como entenderla)

```
a = ALGUIEN_QUE_NO_CUMPLA;
b = ALGUIEN_QUE_CUMPLA;
```

# Búsqueda binaria (como entenderla)

```
a = ALGUIEN_QUE_NO_CUMPLA;
b = ALGUIEN_QUE_CUMPLA;
while (b-a > 1) {
    c = (a+b)/2;
```

# Búsqueda binaria (como entenderla)

```
a = ALGUIEN_QUE_NO_CUMPLA;
b = ALGUIEN_QUE_CUMPLA;
while (b-a > 1) {
    c = (a+b)/2;
    if (c CUMPLE)
       b = c;
    else
       a = c;
}
// Llegados a este punto, a es el ultimo que no cumple, y b es el primero que cumple.
```

# Búsqueda binaria

```
int busquedaBinaria(int arreglo[], int tam, int elemento) {
  a = 0;
  b = tam - 1;
  while (b-a > 1) {
     c = (a+b)/2;
     if (arreglo[c] >= elemento)
        b = c;
     else
        a = c;
  }
  return b;
}
```

# Ejercicios

14) Escribir una función que permita rotar una posición a la izquierda todos los elementos del vector. La rotación es circular (el elemento en la posición cero se desplaza “circularmente” a la última posición del vector). Ejemplo: rotar_izquierda([1,2,3,4], 4) → [2,3,4,1]

# Ejercicios

14) Escribir una función que permita rotar una posición a la izquierda todos los elementos del vector. La rotación es circular (el elemento en la posición cero se desplaza “circularmente” a la última posición del vector). Ejemplo: rotar_izquierda([1,2,3,4], 4) → [2,3,4,1]

```
void rotar_izquierda(int arreglo[], int tam) {
  if (tam > 0) {
    int elementoCero = arreglo[0];
    for (int i = 0; i < tam - 1; i++) arreglo[i] = arreglo[i + 1];
    arreglo[tam - 1] = elementoCero;
  }
  else cout << "No se puede rotar un arreglo de tamaño 0" << endl;
}
```

# Ejercicios

1) Implementar una función que reciba dos listas (arreglos) ordenadas junto con sus respectivos tamaños lógicos y devuelva una nueva lista que contenga todos los elementos de las dos listas originales.

# Ejercicios

1) Implementar una función que reciba dos listas (arreglos) ordenadas junto con sus respectivos tamaños lógicos y devuelva una nueva lista que contenga todos los elementos de las dos listas originales.

```
void combinaListas(int arreglo1[], int tam1, int arreglo2[], int tam2, int arreglo3[], int & tam3) {
  for (int i = 0; i < tam1; i++) {
    arreglo3[tam3] = arreglo1[i];
    tam3++;
  }

  for (int i = 0; i < tam2; i++) {
    arreglo3[tam3] = arreglo2[i];
    tam3++;
  }
}
```

# Ejercicios

2) Una función debe recibir un vector de números enteros y su tamaño lógico (o sea, una lista de enteros). Debe retornar los datos actualizados, insertando un elemento detrás de cada elemento que tenga un valor X múltiplo de 100, con el valor X+1; y además devolver la cantidad de elementos insertados. Suponer que el tamaño físico del arreglo alcanza para hacer las inserciones. No recorrer el arreglo más de una vez.

# Ejercicios

3) Una función debe recibir una lista de números enteros positivos. Debe retornar los datos actualizados, eliminando del vector los valores repetidos; y además devolver la cantidad de elementos pares eliminados y el promedio de los valores que quedan en el vector.

# Ejercicios

4) Una función debe recibir una lista ordenada de números enteros positivos. Debe retornar los datos actualizados, eliminando del vector todos los valores impares que se encuentran en medio de dos valores pares; y además devolver la cantidad de elementos impares eliminados y la mayor diferencia entre dos elementos pares sucesivos considerando los valores que quedan en el vector.