Métodos de Ordenamiento

- Burbujeo – Burbujeo Optimizado
- Selección
- Inserción

BURBUJEO

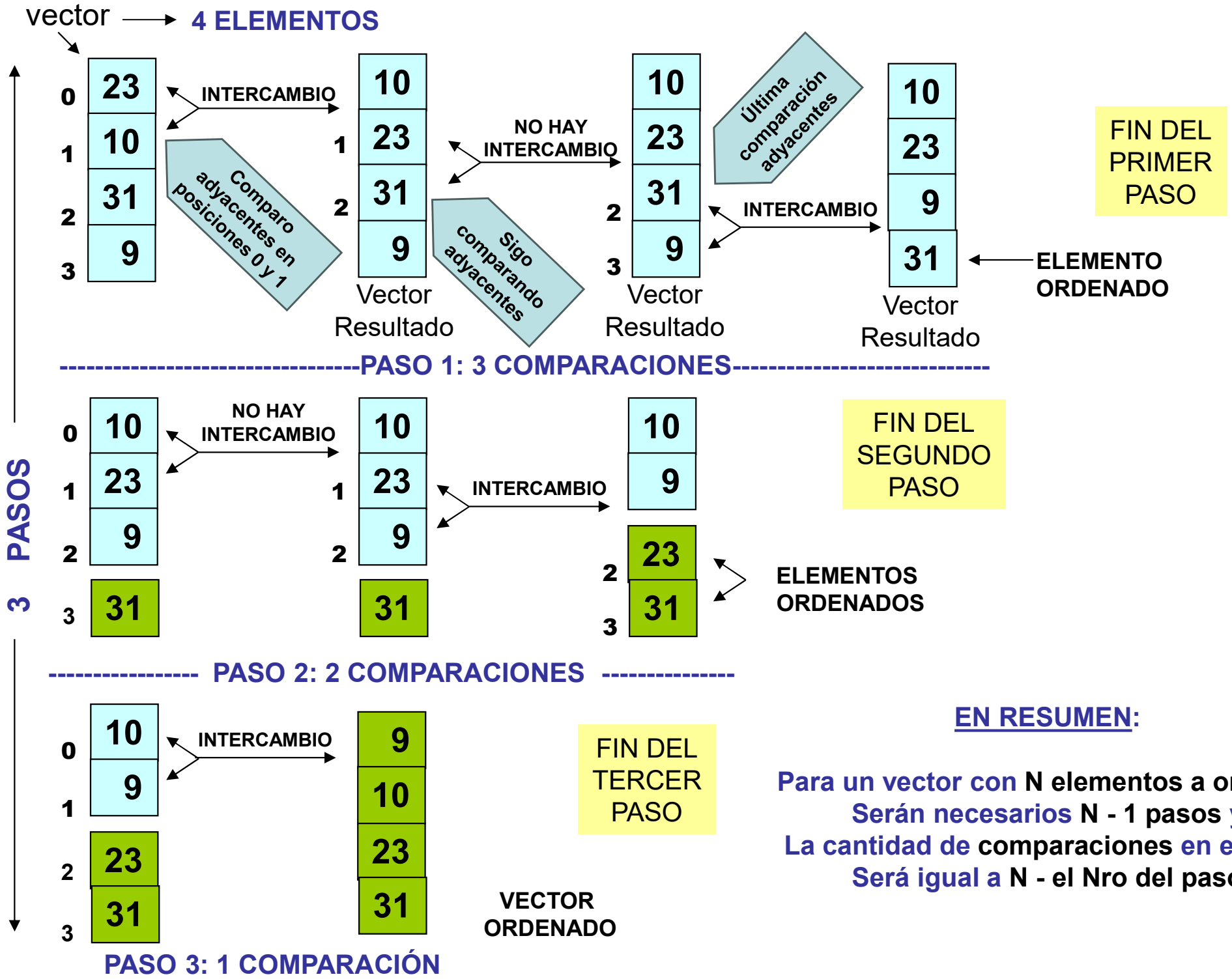
Objetivo:

Ordenar los elementos del vector,

comparando en pasadas sucesivas,

aquellos **elementos** que son **adyacentes**.

BURBUJEO – Desarrollo del proceso



EN RESUMEN:

Para un vector con N elementos a ordenar,
Serán necesarios N - 1 pasos y
La cantidad de comparaciones en el paso
Será igual a N - el Nro del paso

BURBUJEO – Desarrollo del Algoritmo

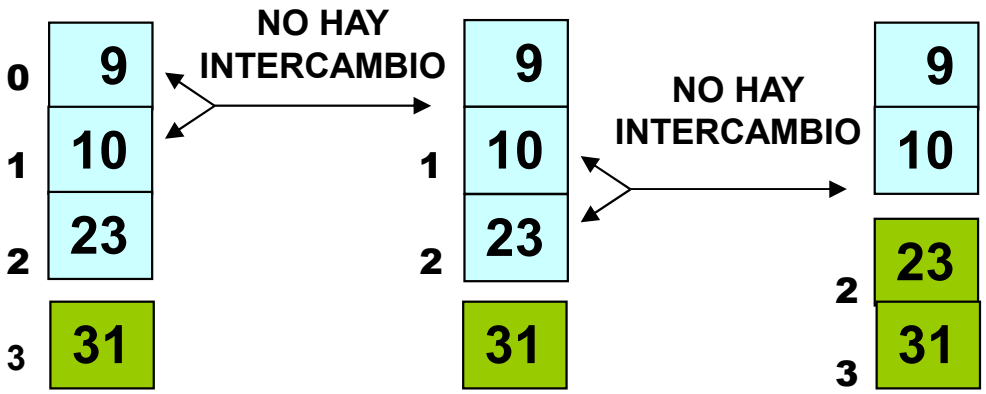
(ver programa en C)

```
void ordenar_por_burbujeo (tvec vec, int ml)
{
    int i, j, aux;

    for (i = 1; i < ml; i++)                // Pasos

        for (j = 0; j < ml - i; j++)        // Comparaciones

            if (vec[j] > vec[j+1])
            {
                aux = vec[j];               // Intercambio
                vec[j] = vec[j + 1];
                vec[j+1] = aux;
            }
}
```

BURBUJEO – Optimización

Qué sucede si en un paso completo, no hubo ningún intercambio?

PASO 2



VECTOR ORDENADO DETENERSE!

AYUDA PARA MODIFICAR EL ALGORITMO:

- DETECTAR SI HUBO O NO ALGUN INTERCAMBIO EN UN PASO
- ANTE LA DETECCION, NO CONTINUAR CON LOS PASOS SIGUIENTES

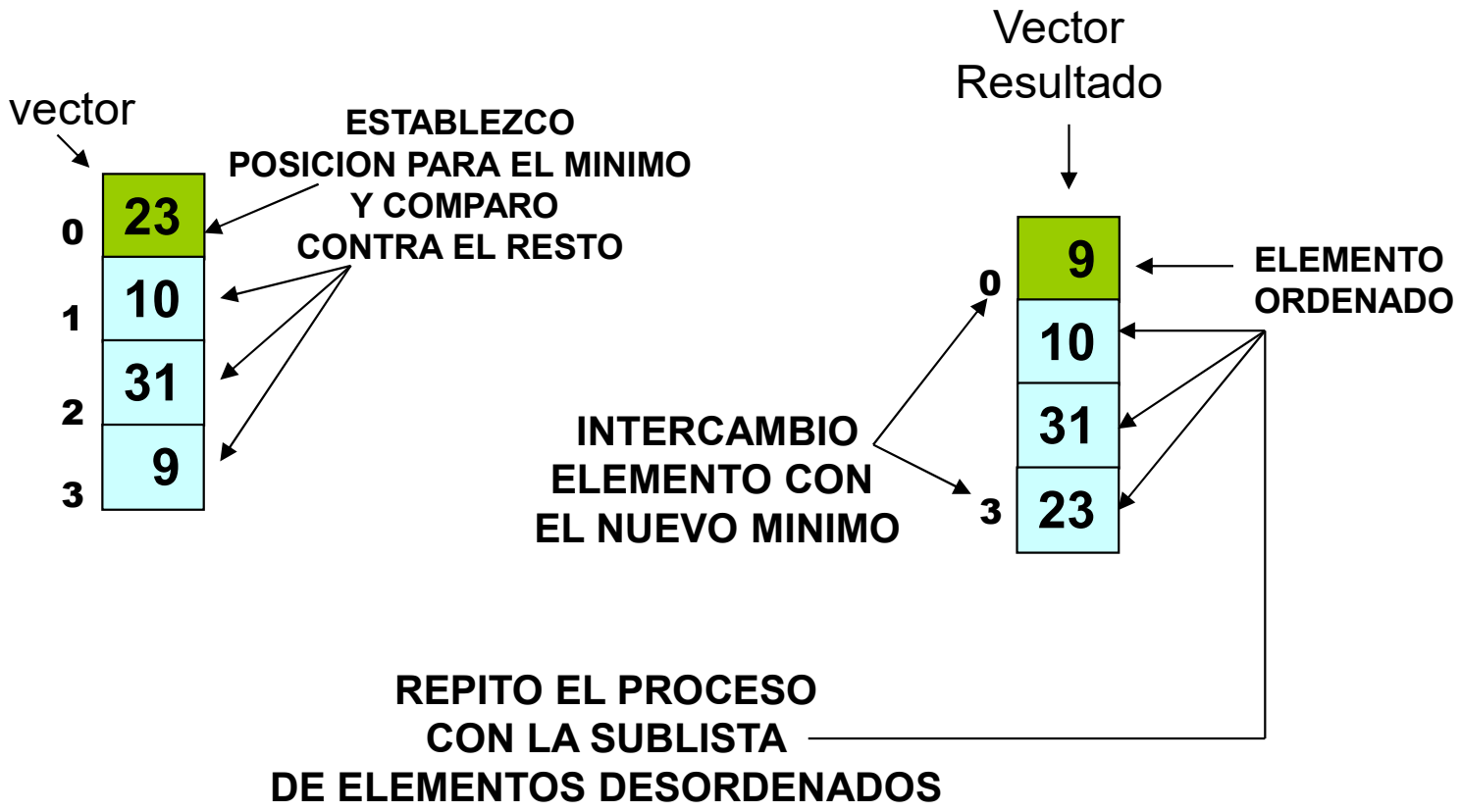
```
void ordenar_por_burbujeo_optimizado (tvec vec, int ml)
{
    int i, j, aux;

    i = 1;

    bool hubo_intercambio = true;

    while ((i < ml) && hubo_intercambio)            // Pasos
    {
        hubo_intercambio = false;
        for (j = 0; j < ml - i; j++)                // Comparaciones

            if (vec[j] > vec[j+1])
            {
                aux = vec[j];                        // Intercambio
                vec[j] = vec[j + 1];
                vec[j+1] = aux;
                hubo_intercambio = true;
            }
        i++;
        }
}
```

SELECCION

Objetivo:

Ordenar los elementos del vector, **buscando el mínimo elemento entre una posición i y la posición final**, e intercambiar el mínimo con el elemento de la posición i.

SELECCION – Desarrollo del proceso



```
void ordenar_por_seleccion (tvec vec, int ml)
{
    int i, j, aux, posMinimo;

    for (i = 0; i < ml-1; i++)
    {
        posMinimo = i;

        for (j = i+1; j < ml; j++)

            if (vec[j] < vec[posMinimo])

                posMinimo = j;

        aux = vec[i];
        vec[i] = vec[posMinimo];
        vec[posMinimo] = aux;

    }
}
```

INSERCION

Objetivo:

Ordenar los elementos del vector, **tomando un elemento y comparándolo contra los anteriores ya ordenados**, deteniéndose cuando se encuentra uno menor y entonces es allí donde se inserta, desplazando el resto de los elementos mayores.

INSERCION

El Algoritmo:

- Vamos a **recorrer** todo el vector A comenzando **desde la posición inicial del vector + 1**, hasta la última posición.

- Para cada elemento A[i], se trata de **ubicar en el lugar correcto el elemento A[i]** en cuestión, entre los elementos anteriores: A[i-1], A[i-2], ..., A[inicial].

- Dada la posición actual i, el algoritmo se basa en que los **elementos A[inicial], A[1], ...,A[i-1] ya están ordenados**.

```
void ordenar_por_insercion (tvec vec, int ml)
{
        int i, j, aux;

    for (i = 1; i <= ml; i++)
    {
        aux = vec[i];                // preservo valor
        j = i - 1;

        while ((j >= 0) && (vec[j] > aux))
        {
            vec[j+1] = vec[j];     // desplazo elemento
            j = j - 1;
        }

        vec[j+1] = aux;

    }
}
```