**Algoritmos de ordenación y búsqueda**

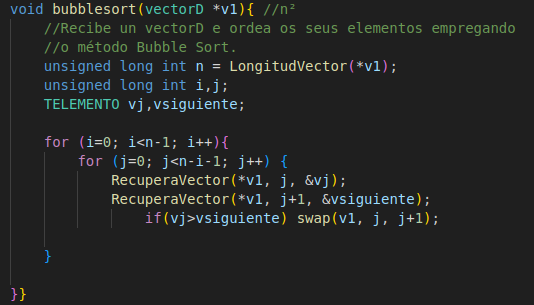
**Tipos de operaciones**

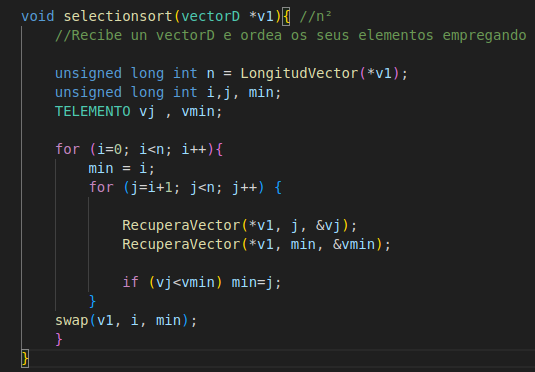
* **Operaciones internas:** Se realizan en memoria principal
* **Operaciones externas:** Datos almacenados en un dispositivo externo.

**Algoritmos de búsqueda**

* **Búsqueda secuencial:** Compara cada elemento del conjunto con el valor deseado hasta encontrarlo o llegar al fin de la lista.
  + Búsqueda con centinela: guarda el elemento deseado al final de la lista, e itera por la lista comparando hasta encontrar el elemento deseado. Si lo encuentra al final, confirma que no está en la lista.
    - Sólo 1 comparación por bucle → más eficiente
  + Complejidad lineal (O(N))
* **Búsqueda binaria:** Requiere una lista ordenada. Se comienza por el central y según sea mayor o menor que el buscado se continúa con la primera o segunda mitad de la lista.
  + Complejidad O(log2N)

**Bubble sort**

* Se recorre la array un cierto número de veces, comparando los pares de valores consecutivos.
  + Si no están ordenados, se intercambian.
* En la primera pasada el mayor elemento estará al final, en la segunda iteración el segundo mayor llegará a la penúltima posición, y así sucesivamente.
* Orden promedio **O(N2)**
* **Burbuja mejorada:** Tras cada iteración, comprueba si se ha realizado algún intercambio. Si se ha pasado por toda la lista sin cambiar nada, se considera ordenada y finaliza el programa.

**Selection sort**

* Se divide la lista en dos partes, una ordenada y otra desordenada (inicialmente, la ordenada está vacía)
* En cada iteración, se selecciona el menor elemento de la lista desordenada y se coloca en su posición correspondiente en la lista ordenada.
* Orden promedio **O(N2)**

**Insertion sort**

* Similar a Selection Sort. En cada iteración, en lugar de tomar el menor elemento de la lista desordenada se toma el primero.
* Cada elemento se coloca en la posición correcta de la lista ordenada (antes o después de los elementos ya ordenados según su valor)
  + Para esto, es necesario hacerle hueco desplazando los elementos necesarios a la derecha.
* Orden promedio **O(N2)**

**Quick sort**

* Utiliza la táctica ‘divide y vencerás’.
* Se escoje un elemento cualquiera (normalmente el primero) como **pivote**. Se colocan todos los elementos menores que el pivote a su izquierda, y los mayores a su derecha.
* Luego, se aplica el mismo método a cada una de las dos mitades del array.
* Método implementado:
  + Se realizan dos búsquedas, una de izquierda a derecha, buscando un elemento mayor que el pivote, y una de derecha a izquierda, buscando un elemento menor que el pivote.
  + Cuando se han encontrado ambos se intercambian, y se sigue realizando la búsqueda hasta que las dos búsquedas se encuentran.