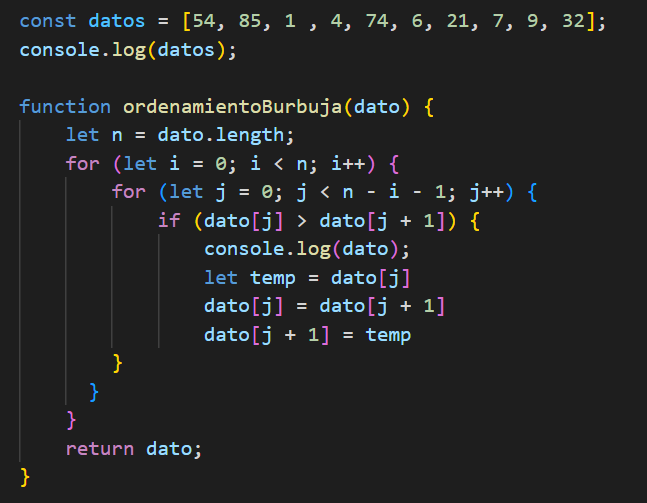
Ana Valeria Mendoza Cipagauta

**Métodos de ordenamiento**

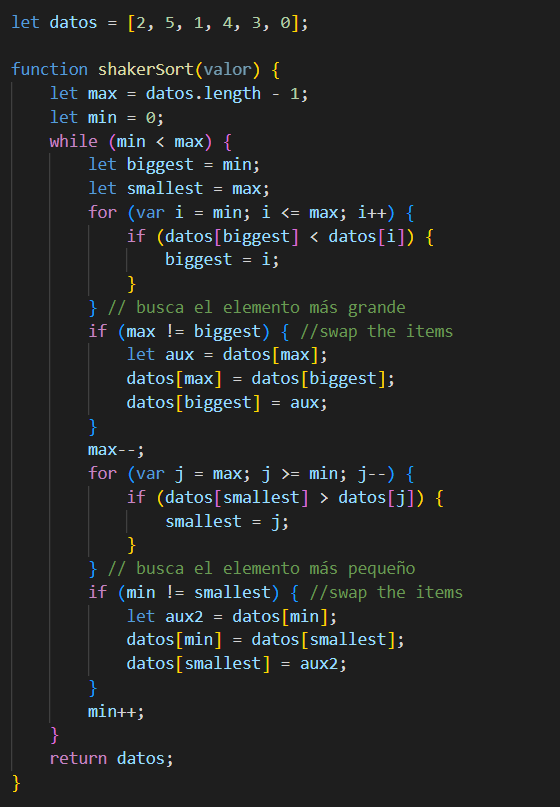
**Ordenamiento Burbuja**



* ***Análisis temporal***  
  En el peor de los casos, los bucles *for* tendrían que realizar *n* comparaciones siendo así, cada *for* equivale a y como hay dos *for* en el algoritmo la complejidad será .
* ***Análisis espacial***
  + El arreglo *dato*s es un arreglo de entrada donde se almacenan los números que se van a ordenar, su tamaño no cambia durante la ejecución del algoritmo, por lo que su espacio en memoria es a , donde a es el tamaño del arreglo.
  + La variable *n* se usa para almacenar la longitud del arreglo *dato*, por lo que su espacio en memoria es a , ya que es una sola variable.
  + Las variables *i* y *j* son usadas en los bucles y ocupan espacio en memoria para almacenar un número entero, pero no crece con el tamaño de la entrada, por lo que su espacio en memoria de cada una es .
  + La variable *temp* se usa para almacenar temporariamente un valor durante el intercambio de elementos en el arreglo y ocupa espacio en memoria para almacenar un número entero, pero su espacio es constante, no crece con el tamaño de la entrada, por lo que su espacio en memoria es .

El espacio de memoria usado en este algoritmo sería por el arreglo donde se almacenan los datos, ya que el resto de las variables ocupan la misma cantidad de espacio durante toda la ejecución.

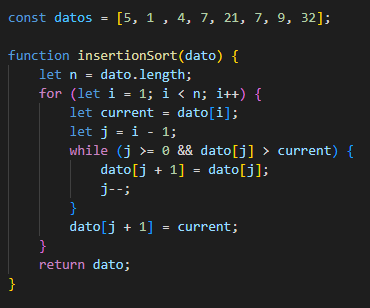
**Ordenamiento shakerSort**



* ***Análisis temporal***  
  En el peor de los casos, el bucle *while* tendrían que realizar *n* veces por ende equivale a . Dentro del bucle de *while* existen 2 bucles *for* cada uno equivale pero como se ejecutan uno a la vez, la complejidad sería de .
* ***Análisis espacial***
  + El arreglo *dato*s es un arreglo de entrada donde se almacenan los números que se van a ordenar, su tamaño no cambia durante la ejecución del algoritmo, por lo que su espacio en memoria es a , donde a es el tamaño del arreglo.
  + La variable *max, min, biggest, mallest* se usa para almacenar valores enteros que su tamaño no aumenta, por lo que su espacio en memoria es a .
  + Las variables *i* y *j* son usadas en los bucles y ocupan espacio en memoria para almacenar un número entero, pero no crece con el tamaño de la entrada, por lo que su espacio en memoria de cada una es .
  + La variable *aux* y *aux2* se usa para almacenar temporariamente un valor durante el intercambio de elementos en el arreglo y ocupa espacio en memoria para almacenar un número entero, pero su espacio es constante, no crece con el tamaño de la entrada, por lo que su espacio en memoria es .

El espacio de memoria usado en este algoritmo sería por el arreglo donde se almacenan los datos, ya que el resto de las variables ocupan la misma cantidad de espacio durante toda la ejecución.

**Ordenamiento insertionSort**

****

* ***Análisis temporal***  
  En el peor de los casos, el bucle *for* se tendrían que realizar *n* veces por ende equivale a . Dentro del bucle de *for* existen un bucle *while* y equivale a por ende la complejidad sería de .
* ***Análisis espacial***
  + El arreglo *dato*s es un arreglo de entrada donde se almacenan los números que se van a ordenar, su tamaño no cambia durante la ejecución del algoritmo, por lo que su espacio en memoria es a , donde a es el tamaño del arreglo.
  + La variable *n* se usa para almacenar la longitud del arreglo *dato*, por lo que su espacio en memoria es a , ya que es una sola variable.
  + Las variables *i* y *j* son usadas en los bucles y ocupan espacio en memoria para almacenar un número entero, pero no crece con el tamaño de la entrada, por lo que su espacio en memoria de cada una es .
  + La variable *current* se usa para almacenar temporariamente un valor durante el intercambio de elementos en el arreglo y ocupa espacio en memoria para almacenar un número entero, pero su espacio es constante, no crece con el tamaño de la entrada, por lo que su espacio en memoria es .

El espacio de memoria usado en este algoritmo sería por el arreglo donde se almacenan los datos, ya que el resto de las variables ocupan la misma cantidad de espacio durante toda la ejecución.