

Uso de estructuras de datos en el programa:

funciones básicas: Muestra la cantidad de letras, líneas, palabras y palabras diferentes

Llama al método load():

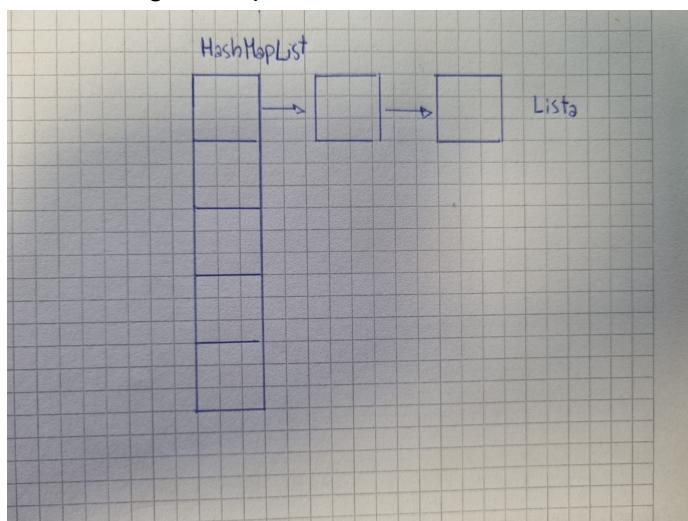
Recorre cada línea del archivo contando las líneas, letras, palabras y palabras diferentes

Carga las palabras en un HashMapList

Muestra la cantidad de letras, líneas, palabras y palabras diferentes.

Funcionamiento de la clase HashMapList:

Un hash de listas que cuando se agrega una palabra la guarda al final de la lista si es que no está cargada la palabra.



-palabras: Muestra las palabras en orden alfabetico

@param HashMapTree<string>*

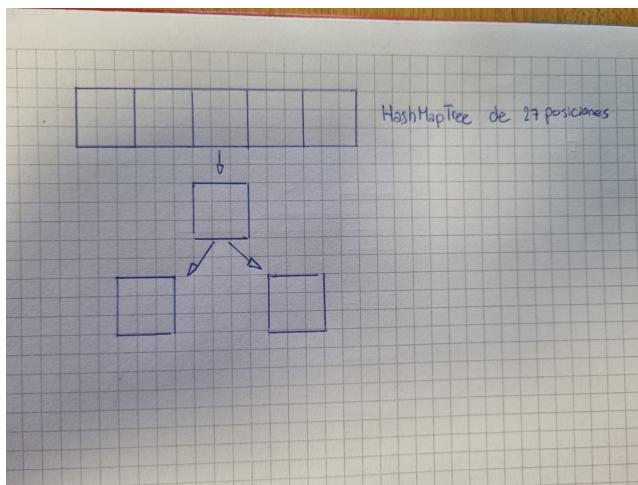
@param int nPalabras

Se recorre un un HashMapTree inorder y se muestran las nPalabras solicitadas. Al recorrerlo inorder las palabras se muestran en orden alfabético.

Funcionamiento de la clase HashMapTree:

Un hash de 27 posiciones, en cada posición tiene un árbol binario. Una posición para cada letra del abecedario, y una posición para los caracteres especiales.

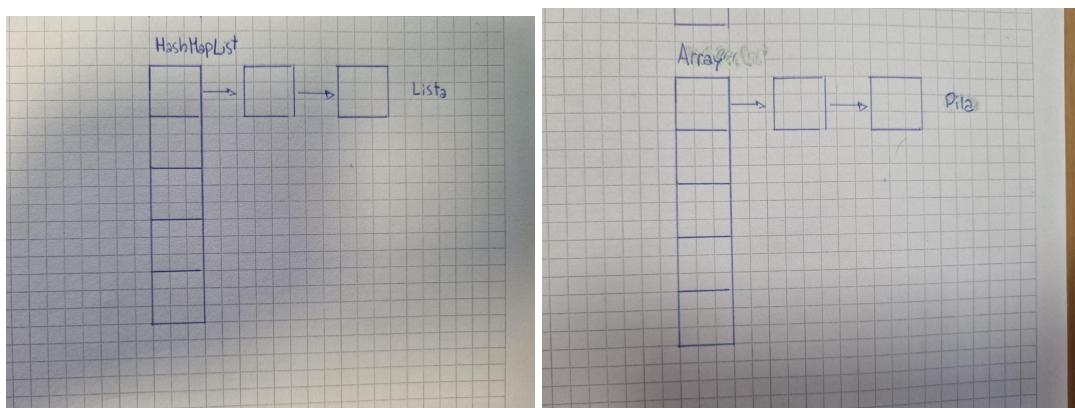
La función hash inserta una palabra en el árbol que esté en la posición del valor ascii de su primer carácter.



-ocurrencias: Muestra las palabras por orden de ocurrencias

```
@param HashMapList<string>*
@param int nPalabras
```

Carga las palabras del texto en un HashMapList para encontrar la ocurrencia de cada palabra. Luego recorre el HashMapList y mientras saca los datos de este los va cargando en un Array de punteros a Pila, donde el tamaño del Array va a ser la mayor ocurrencia y la posición de la palabra en este va a ser su ocurrencia. De esta forma se ordena automáticamente mientras se van cargando las palabras al array. Las palabras con la misma ocurrencia se van almacenando en una Pila. Para imprimir, recorre el Array y por cada posición del Array recorre la Pila asociada a este, hasta que la Pila quede vacía o hasta imprimir la cantidad de palabras deseadas.



-mostrar: Busca las palabras pasadas como parámetro y las muestra ordenadas por ocurrencias

```
@param string
```

Carga las palabras en un HashMapList

Busca las palabras pasadas como parámetros y las guarda en una ColaPrioridad según su ocurrencia.

Desencola la colaPrioridad entonces se muestran las palabras según su ocurrencia.

-excluir y excluirf: Modifica los comandos ocurrencias y palabras haciendo que no muestren las palabras pasadas como argumentos o pasadas en un file.txt.

-palabrasExcluir: Devuelve un *HashMapTree sin las palabras pasadas como parámetros

```
@param string  
@return HashMapTree<string>*
```

Carga un HashMapTree con todas las palabras del archivo.

Lee las palabras pasadas como string y las remueve del HashMapTree.

-palabrasExcluirf: Devuelve un *HashMapTree sin las palabras pasadas en un archivo

```
@param string ARCHIVO_EXCLUIR.TXT  
@return HashMapTree<string>*
```

Carga un HashMapTree con todas las palabras del archivo.

Lee las palabras del archivo y las remueve del HashMapTree.

-ocurrenciasExcluir: Devuelve un *HashMapList sin las palabras pasadas como parámetros

```
@param string  
@return HashMapList<string>*
```

Carga un HashMapList con todas las palabras del archivo.

Lee las palabras pasadas como string y las remueve del HashMapList.

-ocurrenciasExcluirf: Devuelve un *HashMapList sin las palabras pasadas en un archivo

```
@param string ARCHIVO_EXCLUIR.TXT  
@return HashMapList<string>*
```

Carga un HashMapList con todas las palabras del archivo.

Lee las palabras del archivo y las remueve del HashMapList.