Práctica 5. Tablas Hash

Sesiones de prácticas: 2

Objetivos

Implementación y optimización de tablas de dispersión cerrada.

Descripción de la EEDD

Como el Quijote tiene una gran cantidad de palabras, se ha propuesto cambiar el diseño para que las búsquedas de las palabras en el diccionario con verbos sean aún más eficientes. Para ello se ha pensado en utilizar una tabla de dispersión para almacenar las palabras del diccionario y de los verbos conjugados. Para ello se va a utilizar una tabla hash cerrada que contemple además de la búsqueda nuevas inserciones y borrados.

Recordad que las claves en dispersión deben ser de tipo unsigned long, por lo que un string debe ser previamente convertido a este tipo mediante la función djb2().

La tabla de dispersión no se implementará esta vez mediante un template, su definición sigue esta especificación:

- THashPalabra::hash(unsigned long clave, int intento), función privada con la función de dispersión

- THashPalabra::THashPalabra(int tamTabla). Constructor que construye una tabla dado un tamaño predefinido.

- THashPalabra::THashPalabra(THashPalabra &thash). Constructor copia.
- THashPalabra::operator=(THashPalabra &thash).
- ~THashPalabra().

- bool THashPalabra::insertar(unsigned long clave, Palabra &pal), que inserte una nueva palabra en la tabla. No se permiten repetidos por tanto, es necesario buscar antes de insertar.

- bool THashPalabra::buscar(unsigned long clave, string &termino, Palabra &*pal), que busque un dato a partir de su clave numérica y devuelva el objeto Palabra a través de un puntero. Recordad que hay que comprobar que *termino* coincide con la clave buscada por eso se pasa el parámetro *termino* por la cabecera.

- bool THashPalabra::borrar(unsigned long clave, string &termino), que borre la palabra de la tabla. Recordad que hay que comprobar que *termino* coincide con la clave buscada antes de borrar.

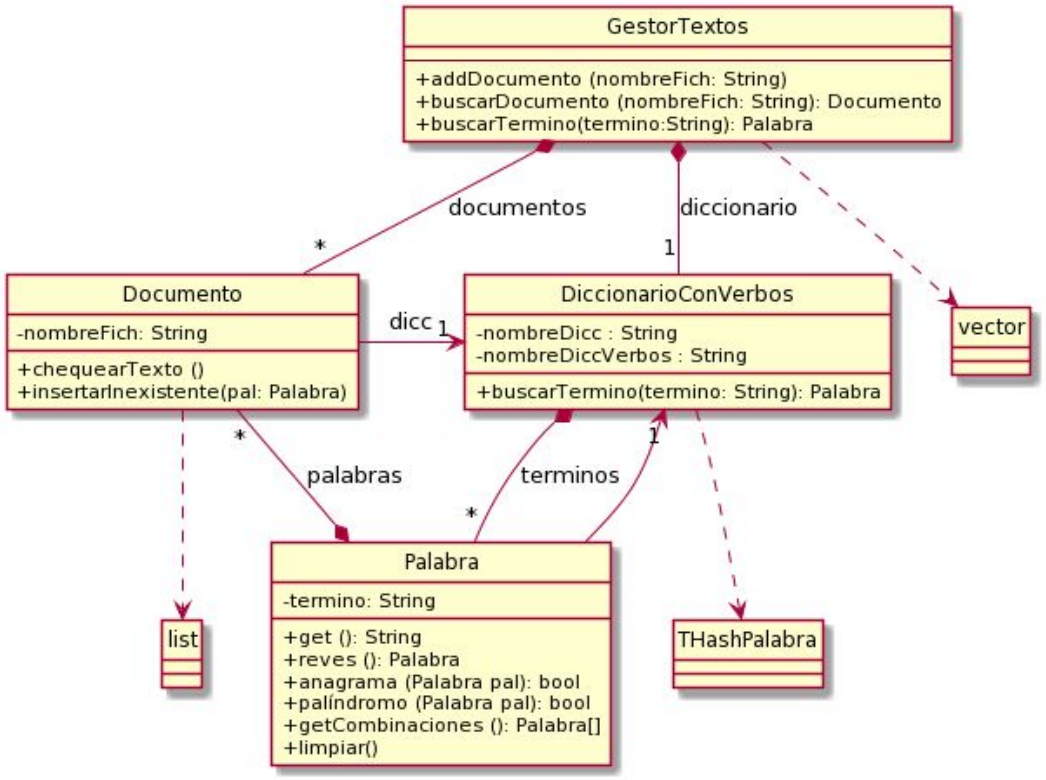
- unsigned int THashPalabra::numPalabras(), que devuelva el número de palabras que contiene el diccionario.

- void THashPalabra::redispersar(unsigned tam), que redispersa la tabla a un nuevo tamaño *tam.* **A implementar sólo por aquellos que trabajan en parejas.**

La clase *GestorTextos*, tiene un vector de documentos y un diccionario y puede realizar la búsqueda de una palabra en el diccionario.

La clase *DiccionarioConVerbos* almacena en una tabla hash las palabras del diccionario y de los verbos conjugados.

La clase *Documento* se sigue encargando de *chequearTexto()*. Se encarga de buscar las palabras del documento en el diccionario y si no están las inserta en una lista de palabras inexistentes.



Programa de prueba 1:

Antes de que la tabla deba ser utilizada, se debe entrenar convenientemente para determinar qué configuración es la más adecuada. Para ello se van a añadir nuevas funciones que ayuden a esta tarea:

- unsigned int THashPalabra::maxColisiones(), que devuelve el número máximo de colisiones que se han producido en la operación de inserción más costosa realizada sobre la tabla.
- unsigned int THashPalabra::promedioColisiones(), que devuelve el promedio de colisiones por operación de inserción realizada sobre la tabla.
- float THashPalabra::factorCarga(), que devuelve el factor de carga de la tabla de dispersión.
- unsigned int THashPalabra::tamTabla(), que devuelve el tamaño de la tabla de dispersión.

Ayudándose de estas funciones, se debe configurar una tabla (en word o similar) que rellene estos valores de máximo de colisiones, factor de carga y promedio de colisiones con **tres funciones hash** y con **dos tamaños de tabla diferentes** considerando un factor de carga $\boldsymbol{\lambda}$

>= 0.6. Para determinar el tamaño de la tabla, hay que obtener el siguiente **número primo** después de aplicar el $\boldsymbol{\lambda}$ al tamaño de los datos.

Se probará una función de dispersión cuadrática y dos con dispersión doble. En total salen 6 combinaciones posibles. En base a estos resultados, se elegirá la mejor configuración para balancear el tamaño de la tabla y las colisiones producidas. **El fichero word debe llamarse *analisis_Thash* y debe subirse junto al proyecto**.

Programa de prueba 2:

1. Insertar una nueva palabra en la tabla hash, *wifi*.
2. Mostrar el número de colisiones que se han producido al insertarla.
3. Borrar de la tabla hash todas aquellas palabras que comiencen con 'w'. Mostrar cuántas palabras se han borrado. Existen unas 20 palabras en el diccionario general, meterlas directamente en un vector para luego extraerlas de la tabla.
4. Insertar de nuevo las palabras, 'waterpolo' y 'windsurf' (de esta forma comprobamos si se eliminaron correctamente en el punto anterior).
5. **Para los que trabajan con parejas**: insertar en la tabla hash todas las palabras que contiene el documento siglas.txt y aplicar redispersión para dejar el valor del factor de carga igual que el valor que tenía antes de insertar las palabras del fichero. Recordad buscar el término antes de insertar, puede ser que un acrónimo coincida con una palabra del diccionario y algunos términos del fichero siglas están repetidos (tienen diferentes significados pero estos han sido borrados del fichero).

Estilo y requerimientos del código:

1. El código debe ser claro, tener un estilo definido y estar perfectamente indentado, para ello se pueden seguir algunos de los estilos preestablecidos para el lenguaje C++ (http://geosoft.no/development/cppstyle.html).

2. Deben comprobarse todas los posibles errores y situaciones de riesgo que puedan ocurrir (desbordamientos de memoria, parámetros con valores no válidos, etc.) y lanzar las excepciones correspondientes, siempre que tenga sentido. Leer el tutorial de excepciones disponible en el repositorio de la asignatura en docencia virtual.

3. Se valorará positivamente la calidad general del código: claridad, estilo, ausencia de redundancias, etc.