Práctica 4. STL de contenedores lineales

Sesiones de prácticas: 1

Objetivos

Manejar los contenedores lineales de STL.

Descripción de la EEDD

Las EEDD que se han estudiado en teoría son similares en su implementación a las de STL y sobre todo en lo relativo a la eficiencia. En esta práctica vamos a mantener la misma funcionalidad que la Práctica 3 pero cambiando nuestras estructuras de datos a contenedores de STL.

En concreto se usará:

- **std::vector** en la relación de Diccionario con Palabra
- **std::list** se usa en la relación *siguientes* entre Palabra y Sucesor y además para devolver la lista de sucesores en la función Palabra::sucesores()
- **std::priority_queue**[1] para obtener los sucesores con más ocurrencias

La funcionalidad no se va a cambiar, pero se mejorará el proceso de gestión de sucesores para una palabra utilizando una *priority_queue*. Se construirá una prioriy_queue a partir de la lista de sucesores para considerar como más prioritario los el Sucesor que más número de ocurrencias tenga.

Para ello dotar a *Sucesor* del *operator<* para dar más prioridad al de mayor número de ocurrencias. Para obtener los diez más prioritarios basta con sacarlos con la operación *pop()* ya que el *priority_queue* no permite la iteración por sus datos.

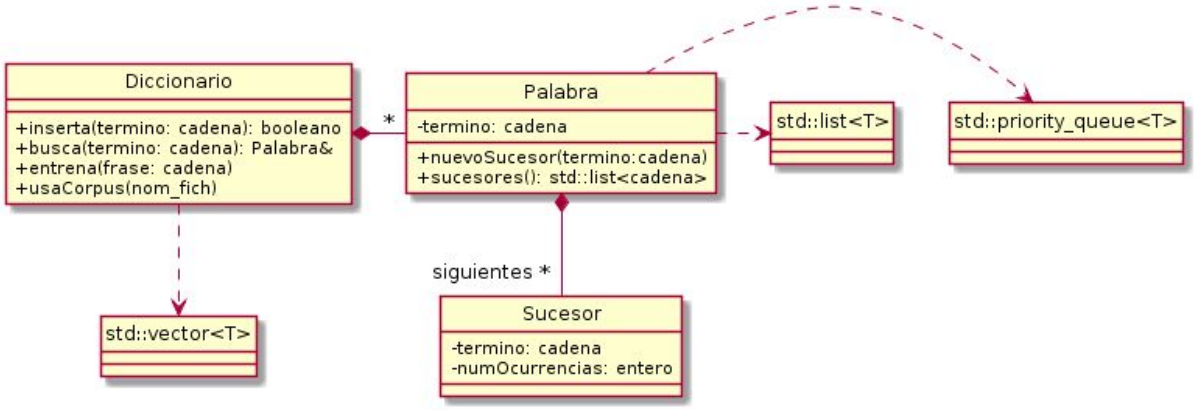
[1] http://www.cplusplus.com/reference/queue/priority_queue/

Programa de prueba: mostrar los términos más habituales

El programa de prueba ahora mostrará los diez términos más prioritarios, en caso de que existan. Se podrán seleccionar del mismo modo que la práctica anterior, eligien do el número asociado a cada uno de ellos que en este caso indicará el orden por prioridad.

Diseño

El diseño es similar al de la práctica 3:



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.