Práctica 2. Implementación de una lista dinámica mediante plantillas y operadores en C++

Sesiones de prácticas: 2

Objetivos

Implementar y utilizar la clase `ListaEnlazada<T>` y su clase auxiliar de tipo iterador ListaEnlazada<T>::`Iterador` utilizando **patrones de clase y excepciones**. Programa de prueba para comprobar su correcto funcionamiento.

Descripción de la EEDD

Implementar la clase `ListaEnlazada<T>` para que tenga toda la funcionalidad de una lista simplemente enlazada en memoria dinámica descrita en la Lección 6, utilizando patrones de clase y excepciones. Los métodos a implementar serán los siguientes:

- Constructor por defecto `ListaEnlazada<T>()`
- Constructor copia `ListaEnlazada<T>(const ListaEnlazada<T>& origen)`.
- Operador de asignación (=)
- Obtener los elementos situados en los extremos de la lista sin modificar la lista: `T& inicio()` y `T& Fin()`
- Obtener un objeto iterador que permita iterar sobre la lista: `ListaEnlazada<T>::Iterador iterador()` e implementar la funcionalidad completa del iterador.
- Insertar en *O(1)* por ambos extremos de la lista, `void insertaInicio(T& dato)` y `void insertaFin(T& dato)`.
- Insertar en *O(n)* un dato en la posición anterior apuntada por un iterador: `void insertaDelante(Iterador &i, T &dato)`.
- Insertar en *O(1)* un dato en la posición siguiente apuntada por un iterador: `void insertaDetras(Iterador &i, T &dato)`
- Borrar el elemento situado en cualquiera de los extremos de la lista, `void borraInicio()` y `void borraFinal()`. La operación de borrado por el final es lineal.

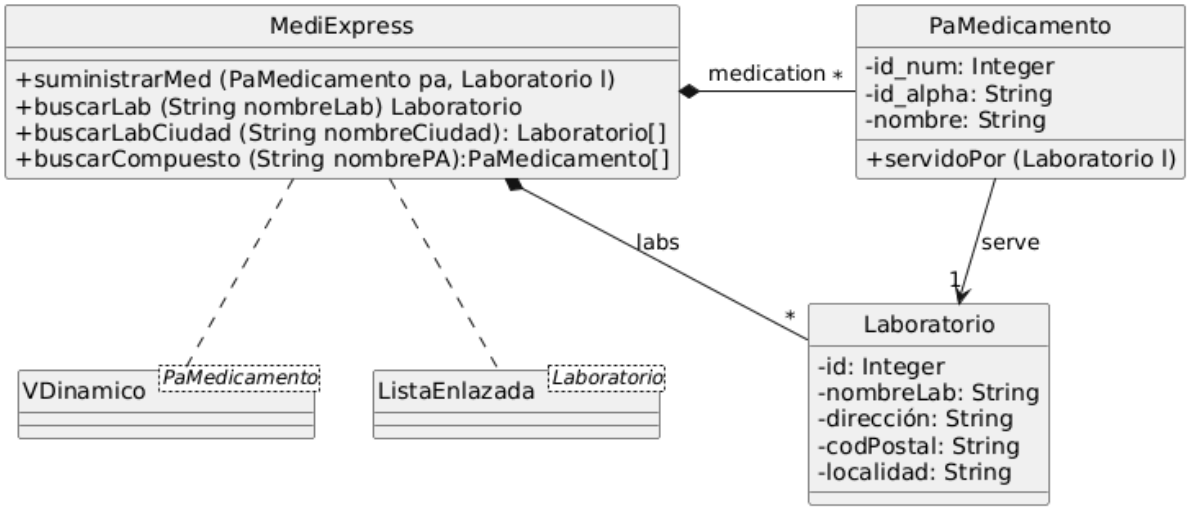
- Borrar el elemento referenciado por un iterador (operación en *O(n)*): `void borra(Iterador &i)`
- `tam(): entero`, que devuelve de forma eficiente el número de elementos de la lista. Se pueden contar los elementos, pero es más eficiente llevar un contador local.
- `concatena(const ListaEnlazada<T> &l):ListaEnlazada<T>`, que devuelve una nueva lista con la concatenación de la lista actual (this) y la proporcionada por parámetro. Sobrecargar también el *operador+* para realizar la misma funcionalidad.
- El destructor correspondiente.

Proyecto de prueba: Gestión de medicamentos Express

MediExpress (Servicio de Medicamentos Express) es una empresa que suministra productos farmacéuticos de forma rápida y eficaz. Para ello, empezaremos con la funcionalidad que se detalla en el siguiente UML. La clase *PaMedicamento* ya se introdujo en la Práctica 1 y corresponde al principio activo de los medicamentos. La nueva clase Laboratorio representa a todos los laboratorios que están registrados en nuestro país. Cada una de estas clases guarda el nombre del laboratorio (*nombreLab*), la dirección donde hace su actividad (*dirección*), el código postal (*codPostal*) y la localidad (*localidad*). Por el momento, cada principio activo es suministrado por un laboratorio mediante la relación *PaMedicamento::serve,* y se enlaza mediante la función *PaMedicamento::servidoPor()*. La clase MediExpress es la clase principal que representa a la nueva empresa. Guarda los principios activos en un vector dinámico, como en la práctica anterior, y los laboratorios en una lista simplemente enlazada.
Por el momento, la funcionalidad de la clase *MediExpress* es básica, se va a encargar de hacer algunas búsquedas y enlazar las clases:

- *suministrarMed()* encargada de enlazar el medicamento con el laboratorio que lo suministra.
- *buscarLab()* que devuelve el laboratorio con el nombre o subcadena indicado, en caso de que exista. Ejem: “BAYER”
- *buscarLabCiudad()* que devuelve todos los laboratorios que se encuentran en la ciudad dada (la búsqueda es por subcadena).
- *buscarCompuesto()* que devuelve todos aquellos principios activos con un nombre dado como subcadena (ya implementado en la Pr1).

IMPORTANTE: no se deben devolver objetos copia



Programa de Prueba I: probar la lista enlazada con enteros

- Implementar la EEDD *ListaEnlazada<T>* y el Iterador<T> con la funcionalidad señalada arriba y de acuerdo con la especificación de la Lección 6.

Probar la robustez de la lista implementando la siguiente funcionalidad:

- Crear una lista de enteros inicialmente vacía.
- Insertar al final de la lista los valores crecientes desde 101 a 200. Mostrar la lista.
- Insertar por el comienzo de la lista los valores decrecientes desde 98 a 1. Mostrar la lista.
- Insertar el dato 100 delante del 101. Mostrar la lista.
- Insertar el dato 99 detrás del 98. Mostrar la lista.
- Borrar de la lista los 10 primeros y los 10 últimos datos. Mostrar la lista.
- Borrar de la lista todos los múltiplos de 10. Mostrar la lista.

Programa de prueba II: probar la funcionalidad de MediExpress

A continuación se describen los pasos que debe seguir el constructor de la clase MediExpress, que tomará los nombres de los ficheros como parámetros:

1. Leer el fichero *"pa_medicamentos.csv"* para cargarlos en el vector dinámico *MediExpress::medication* (similar en Pr1).
2. Adaptar el código de lectura del fichero anterior para leer el nuevo fichero *"laboratorios.csv"*. Durante el proceso de lectura los datos se insertarán en la lista simplemente enlazada en orden por el *id* definida como *MediExpress::labs*. El identificador de cada usuario es el *id*.
3. Enlazar de forma automatizada los medicamentos (un total de 3310) con los laboratorios (un total de 1579). Se hará de manera que cada dos objetos de tipo

PaMedicamento se enlazan con un laboratorio de forma consecutiva (con PaMedicamento::suministrarMed()). De este modo, los dos primeros objetos del tipo *PaMedicamento* se enlazan con el primer laboratorio, el tercer y cuarto con el segundo laboratorio, etc. Con esta secuencia, no todos los 3310 medicamentos se pueden enlazar, quedando 152 principios activos sin laboratorio asociado (1579*2 + 152 = 3310).

Una vez que la estructura está inicializada realizar las siguientes acciones:

1. Buscar y mostrar en pantalla todos los laboratorios ubicados en Granada o provincia
2. Indicar cuántos laboratorios hay en Jaén.
3. Indicar cuántos laboratorios hay en Madrid y mostrar los 10 primeros.
4. Indicar y mostrar en pantalla los laboratorios que suministran todos los productos que sean "ACEITES" (no mostrar los laboratorios repetidos si fuera el caso).
5. Hacer que los 152 medicamentos sin suministradores sean suministrados por los primeros 152 laboratorios ubicados en Madrid de forma consecutiva.

Para los que trabajan en parejas:

1. Eliminar todos los laboratorios de Bruselas, con especial cuidado para que los medicamentos que suministraban ya no los enlacen, es decir, apunten a *nullptr*.

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.