Práctica 3. Árboles AVL

Sesiones de prácticas: 2

Objetivos

Implementar la clase `AVL<T>` utilizando **patrones de clase y excepciones**. Programa de prueba para comprobar su correcto funcionamiento.

Descripción de la EEDD

Implementar la clase `AVL<T>` para que tenga toda la funcionalidad de un árbol equilibrado AVL en memoria dinámica, tal y como se describe en la Lección 11, utilizando patrones de clase y excepciones. Los métodos a implementar serán los siguientes:

En concreto, se usará:

- Constructor por defecto `AVL<T>()`
- Constructor copia (proceso en preorden) `AVL<T>(const AVL<T> &origen)`.
- Operador de asignación (=)
- Rotación a derechas dado un nodo `rotDer(Nodo<T>* &nodo)`
- Rotación a izquierdas dado un nodo `rotIzq(Nodo<T>* &nodo)`
- Operación de inserción `bool inserta(T& dato)`
- Operación de búsqueda recursiva `T* buscaRec(T &dato)`
- Operación de búsqueda iterativa `T* buscaIt(T &dato)`
- Recorrido en inorden[1] `VDinamico<T*> recorreInorden()`
- Número de elementos del AVL `unsigned int numElementos()`
- Altura del AVL, `unsigned int altura()`
- Destructor correspondiente (proceso en postorden)

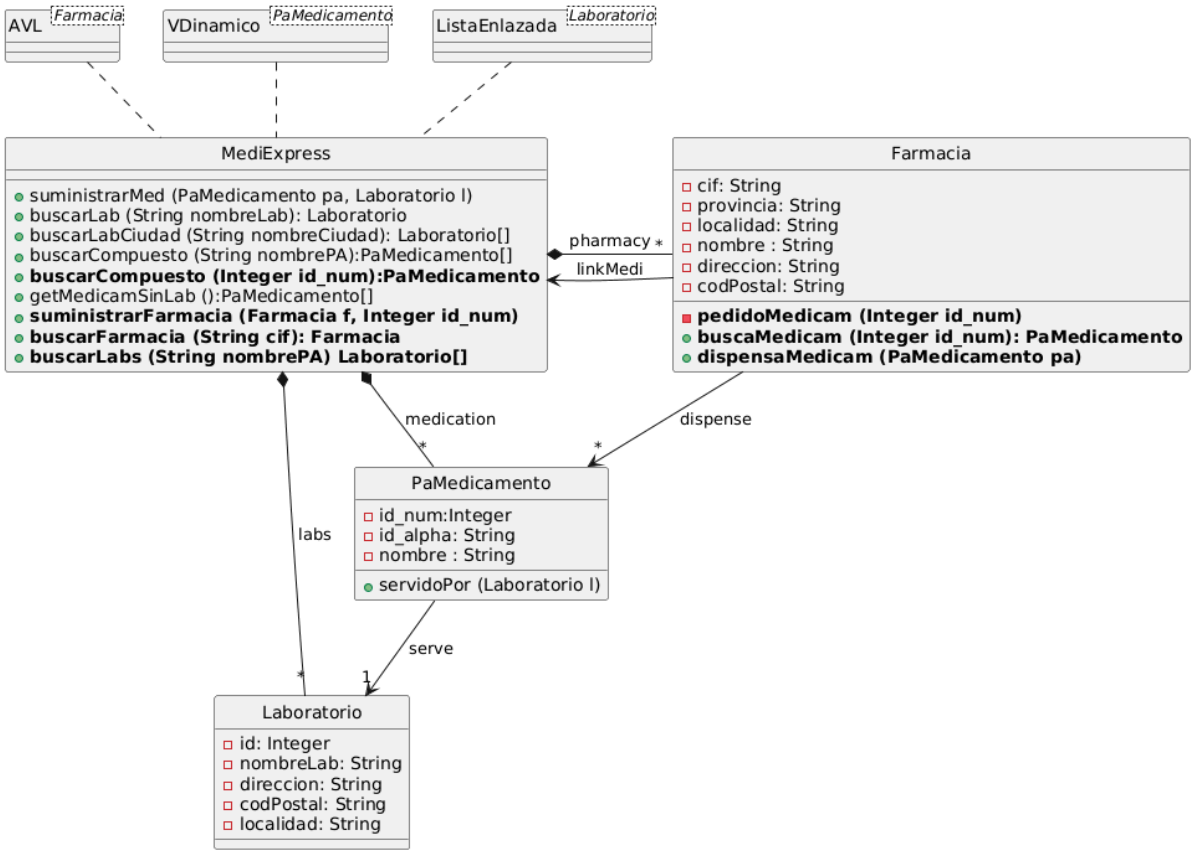
Tanto el **constructor de copia** como el **operador de asignación** deben crear copias idénticas. El constructor se crea mediante un recorrido en **preorden** y el destructor se hace en **postorden**. El contador del **número de elementos** puede realizarse mediante un atributo contador o mediante un recorrido en O(n).

[1] Devuelve un vector dinámico de punteros a los elementos del AVL haciendo un recorrido en Inorden

Proyecto de prueba: Gestión de medicamentos Express

En esta práctica vamos a trabajar también con farmacias, las cuales se abastecen de los medicamentos de forma eficiente a través de *MediExpress*. La nueva funcionalidad va a permitir que las farmacias asociadas con MediExpress hagan sus pedidos a través de la asociación *Farmacia::linkMedi*. De este modo, cuando una farmacia va a vender un medicamento, primero comprueba si lo tiene disponible con *Farmacia::buscaMedicam()*, que busca el medicamento a través de su id numérico a través de la relación *Farmacia::dispenses*, que devuelve el medicamento en caso de que esté. Si no lo tuviera (devuelve nullptr), debe pedir dicho medicamento a *MediExpress* mediante la función *Farmacia::pedidoMedicam()*. Esta función, a través de *Farmacia::linkMedi*, llama a *MediExpress::suministrarFarmacia()*, que lo primero que hace es localizar el medicamento a través de la función *MediExpress::buscarCompuesto()* dando el id del medicamento, y una vez obtenido ya puede enlazar ese medicamento en la farmacia a través de la función *Farmacia::dispensarMedicam()*. La función *MediExpress::buscarFarmacia()* localiza una farmacia dado su CIF y finalmente la función *MediExpress::buscarLabs()* encuentra los laboratorios que suministran el medicamento dado como parámetro (búsqueda parcial por cadena).

El UML de la nueva funcionalidad se refleja a continuación:



Programa de Prueba I: probar la búsqueda de medicamentos de forma eficiente

- Implementar el árbol *AVL<T>* según la especificación de la Lección 11.
- Adaptar el código de la lectura de ficheros anteriores para leer todas las farmacias y añadirlas al árbol AVL del fichero *"farmacias.csv"*.
- Leer de nuevo todas las farmacias y añadirlas también a un vector dinámico.
- Leer de nuevo las 500 primeras farmacias y guardar el CIF en un vector auxiliar (puede ser VDinamico<T> o un buffer).
- Buscar el CIF de esas 500 farmacias en el AVL y contabilizar el tiempo que se tarda en hacer esa búsqueda completa.
- Hacer la misma búsqueda lineal de los 500 datos en el vector dinámico y contabilizar el tiempo que se tarda en hacer esa búsqueda completa.
- Mostrar los tiempos en pantalla y determinar si se ha mejorado el tiempo de búsqueda con el uso del AVL.
- Mostrar la altura del árbol AVL con todas las farmacias.
- Recorrer el árbol en Inorden() y mostrar las 100 primeras farmacias en pantalla.

Programa de prueba II: probar la nueva funcionalidad de MediExpress

A continuación se describen los pasos que debe seguir el **constructor de la clase MediExpress**, que tomará los nombres de los 3 ficheros como parámetros:

1. Leer el fichero *"pa_medicamentos.csv"* para cargarlos en el vector dinámico como hasta ahora.
2. Leer el fichero *"laboratorios.csv"* y ubicar los datos en la lista enlazada, tal y como también se hizo anteriormente.
3. Adaptar el código de la lectura del fichero de farmacias de la Prueba I para que se almacene a través de la relación *MediExpress::pharmacy.*
4. Al igual que en la práctica anterior, enlazar de forma automatizada los medicamentos (un total de 3310) con los laboratorios (un total de 1579), incluyendo a los 152 ubicados en Madrid.
5. Leer de nuevo el fichero de farmacias y guardar sus CIF en un vector. Usar dicho vector para buscar todas las farmacias con *MediExpress::buscarFarmacia()*. Para cada farmacia localizada, asociarle distintos medicamentos mediante la función *MediExpress::suministrarFarmacia() y* de forma consecutiva 100 medicamentos, según el orden del vector y reiterando de forma cíclica. De este modo, la primera farmacia toma los medicamentos del 0 al 99 del vector, la segunda del 100 a 199 y así sucesivamente repitiendo los medicamentos de forma cíclica. Es importante que este enlace se haga correctamente, para ello no crear objetos copia.

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

1. Crear un vector tipo buffer con los siguientes CIF de farmacia: cif = {“37656422V”, “46316032N”, “77092934Q”, “33961602D”, “B62351861”, “B62351861”, “B65828113”, “46138599R”, “35069965W”, “37579913Y”, “37682300C”, “37643742X”, “46112335A”, “47980171D”, “38116138D”, “46315600V”, “37640233C”, “37931842N”, “33964303L”, “35022080A”, “B66046640”, “E66748344”, “47640201W”, “B66621954”, “46121385Z”, “X6806622W”, “46046390E”}.
2. Para todas las farmacias anteriores, buscar si alguna de ellas dispensa el “ÓXIDO DE MAGNESIO” con ID=3640, y si no lo venden, hacer el pedido correspondiente.
3. Buscar y contar todos los laboratorios que trabajen algún tipo de “MAGNESIO”.

Para los que trabajan en parejas:

1. Añadir una función a la clase Farmacia para que localice los medicamentos por nombre (parcialmente). Usarla para localizar todos los laboratorios que suministran a alguna de estas farmacias algún tipo de medicamento con “VIRUS”.

IMPORTANTE:

- **Cuidado con realizar copias de objetos**
- **Añadid *getters* y *setters* cuando se necesiten**