Universidad de Jaén
Departamento de Informática

Prácticas de Estructuras de Datos
Grado en Ingeniería en Informática
Curso 2025/2026

Práctica 1. Implementación de vector dinámico mediante plantillas y operadores en C++

Sesiones de prácticas: 2

Objetivos

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

Descripción de la EEDD

Implementar la clase `VDinamico<T>` para que tenga toda la funcionalidad del vector dinámico descrita en la Lección 4, utilizando patrones de clase y excepciones. Los métodos a implementar serán los siguientes:

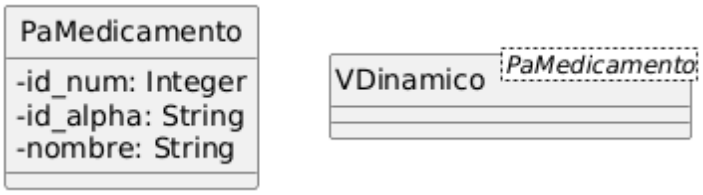
- Constructor por defecto `VDinamico<T>`, iniciando el tamaño físico a 1 y el lógico a 0.
- Constructor dando un tamaño lógico inicial, `VDinamico<T>(unsigned int tamlog)`, empezando el tamaño físico a la potencia de 2 inmediatamente superior a *tamlog* (tamaño lógico).
- Constructor copia `VDinamico<T>(const VDinamico<T>& origen)`.
- Constructor de copia parcial `VDinamico<T>(const VDinamico<T>& origen, unsigned int posicionInicial, unsigned int numElementos)`. En este constructor hay que tener en cuenta que el vector que se genera debe tener un tamaño físico potencia de 2.
- Operador de asignación (=).
- Operador [] para acceder a un dato para lectura/escritura.
- Insertar un dato en una posición: `void insertar(const T& dato, unsigned int pos = UINT_MAX)`. Si no se indica la posición (valor `UINT_MAX`) entonces la inserción se realiza al final del vector. Esta operación siempre incrementa el tamaño lógico en 1.
- Eliminar un dato de una posición intermedia en *O(n)*: `T borrar (unsigned int pos = UINT_MAX)`. Si no se indica la posición (valor `UINT_MAX`) entonces se elimina el último dato del vector.
- Ordenar el vector de menor a mayor. Se puede utilizar la función sort de <algorithm>: `void ordenar()`. Para ello hay que dotar a la clase T del **operator<** (en este caso la clase PaMedicamento debe tener el operador implementado)**.**
- `unsigned int tamlog()` para obtener el tamaño (lógico) del vector.
- El destructor correspondiente.

Programa de prueba: Creación de un vector dinámico de productos farmacéuticos

En este curso vamos a realizar parte de la logística para la gestión y reparto de productos farmacéuticos. Por el momento vamos a trabajar con medicamentos, más concretamente con sus principios activos.

En esta primera práctica utilizaremos la clase `PaMedicamento` representando cada uno de los principios activos de los medicamentos aprobados por la Agencia Española de Medicamentos y Productos Sanitarios (AEMPS). Estos datos se encuentran disponibles en el fichero **pa_medicamentos.csv (pa_medicamentos_mac.csv para usuarios de mac)** adjunta al enunciado (una línea por medicamento). Estos datos se almacenarán en un **vector dinámico**, concretamente, en el contenedor `VDinamico<PaMedicamento>`. Los valores de los atributos aparecen en el fichero en el mismo orden que en el UML, y corresponden con los tres atributos de la clase. El código base de lectura de fichero se proporciona junto al enunciado de la práctica en el fichero **main.cpp**.

La clase `PaMedicamento` contendrá exclusivamente los **constructores**, **operadores** y **getters/setters** necesarios para la práctica. El **criterio de ordenación** de `PaMedicamento` será **por id_num**.



La prueba implementada en la función `main` () consistirá en:

- Implementar el vector dinámico `VDinamico<T>` y la clase `PaMedicamento` atendiendo a las especificaciones anteriores de las Lecciones 3 y 4 del temario.
- Instanciar el vector con todos los objetos de tipo `PaMedicamento` contenidos en el fichero adjunto: **pa_medicamentos.csv**. Mostrar los identificadores de los primeros 50 elementos del vector.
- Ordenar el vector de menor a mayor (por el id_num) y mostrar los datos de los primeros 50 medicamentos.
- Una vez ordenado el vector, buscar en O(*log* n) los medicamentos con los identificadores: 350, 409, 820, 9009 y 12370, mostrando su posición en el vector, teniendo en cuenta que pueden no existir.
- La industria farmacéutica utiliza aceites vegetales y otros aceites comestibles como medicamentos. Localizar mediante búsqueda secuencial todos aquellos medicamentos con la componente “aceite” en cualquier posición del nombre (búsqueda de subcadenas), añadir sus direcciones de memoria (punteros) a otro nuevo vector dinámico mediante la siguiente función y mostrar la información también por pantalla. El compuesto concreto se indica por parámetro (implementar directamente sobre el vector, la función de búsqueda secuencial no está disponible en VDinamico<T>):

```
VDinamico<PaMedicamento*> buscarCompuesto(String &comp,
VDinamico<PaMedicamento> &vMedicamentos);
```

- Aquellos que hagáis las prácticas **por parejas** debéis además:
    - Ordenar el vector por el string "nombre" directamente en el código de prueba con el método burbuja (no incluir esta operación en VDinamico<T>). Mostrar los primeros 50 elementos.
    - Una vez ordenados, averiguar cuántos medicamentos tienen **la primera palabra de su nombre repetida al menos una vez.** Por ejemplo, contabilizamos el ACETATO porque aparece 5 veces, pero no el MANITOL porque aparece solamente una vez. El resultado será un valor numérico que aparecerá por pantalla.

ACETATO CALCIO
ACETATO MAGNESIO TETRAHIDRATO
ACETATO POTASIO
ACETATO SODIO
ACETATO SODIO TRIHIDRATO

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.