Prácticas de Estructuras de Datos
Grado en Ingeniería en Informática
Curso 2022/2023

Práctica 1. Introducción a los contenedores: vectores estáticos y dinámicos (II)

Sesiones de prácticas: 1

Introducción

En la práctica anterior comentamos como es habitual el uso de contenedores de elementos en aplicaciones, por lo que interesa disponer de contenedores de uso general que se adapten fácilmente a diferentes contextos según el uso que se quiera hacer de ellos. En esta segunda parte de la práctica, introduciremos algunas técnicas de diseño de implementación de contenedores de uso general.

Por lo que hemos estudiado en la lección 4, el contenedor implementado en la sesión práctica anterior sería un ejemplo simple de abstracción de vector estático. Sin embargo, dentro de las posibles implementaciones de un contenedor, una de las más habituales son los vectores dinámicos. Estos intentan mejorar las funcionalidades de los vectores básicos de C tales como incrementar (o decrementar) su tamaño máximo a lo largo de la ejecución de la aplicación gestionando de forma eficiente la memoria requerida para ello.

Además, veremos como en los lenguajes con tipos, las plantillas, patrones, o *templates* en inglés, son un recurso sintáctico muy cómodo para adaptar un contenedor de forma que permita utilizarse con diferentes tipos de datos.

Objetivos

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

Descripción de la EEDD

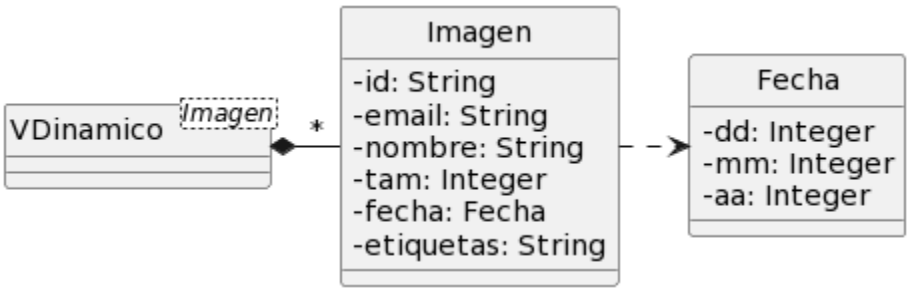
A partir de la clase `ContenedorImagen` implementada en la sesión anterior, implementar la clase `VDinamico<T>` para que tenga toda la funcionalidad del vector dinámico descrita en la Lección 4, utilizando patrones de clase y excepciones. Los métodos a implementar serán los siguientes:

- 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)`, iniciando 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()`.
- Ordenar el vector de mayor a menor con la función `void ordenarRev()`.
- Buscar un dato en el vector utilizando el método de búsqueda binaria o dicotómica y devolviendo la posición del dato. `int busquedaBin(T& dato).` Se crea un objeto básico T que contenga el atributo objeto de la búsqueda que se rellena si la búsqueda tiene éxito. Sino devuelve -1.
- `unsigned int tamlog()` para obtener el tamaño (lógico) del vector.
- El destructor correspondiente.

Programa de prueba: Gestión de un vector dinámico de Imágenes

Con la nueva EEDD vector dinámico, la instanciaremos para gestionar nuestras imágenes de forma que podamos añadir dinámicamente imágenes sin preocuparnos del tamaño que este necesite. Para ello, almacenaremos los datos desde el fichero de la práctica anterior en el nuevo vector dinámico de imágenes. El diagrama UML de la práctica con la nueva clase quedaría de la siguiente forma:



Crear un proyecto independiente con una nueva versión del programa de prueba de la sesión anterior para que realice las mismas operaciones y algunas nuevas trabajando con todos los datos leídos del fichero y almacenados en el nuevo vector dinámico :

- Después de implementar la plantilla VDinamico<T>, modificar el programa de la práctica anterior para almacenar la imágenes del fichero sobre un contenedor de tipo VDinamico.

- Ordenar el contenedor al revés, es decir, de mayor a menor y mostrar los identificadores de las primeras 50 imágenes.
- Ordenar el vector de menor a mayor y mostrar los identificadores de las primeras 50 imágenes.
- Una vez ordenado el vector, buscar imágenes con los identificadores 346335905, 999930245, 165837, 486415569 y 61385551, mostrando su posición en el contenedor. Teniendo en cuenta que pueden no existir.
- El usuario magdalen_upton99@gmail.com desea descargarse y eliminar sus imágenes del sistema. Pasar y borrar todas sus fotos del vector dinámico a un nuevo vector dinámico específico para enviárselas. Mostrar el tamaño lógico de ambos vectores y toda la información de las primeras 10 imágenes que se le enviarán. Si se usan vectores auxiliares, deberán declararse como VDinamico.
- Aquellos que hagáis las prácticas por parejas tendréis que estudiar los tiempos de los cuatro apartados anteriores. Para el resto es opcional.

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.