Universidad de Jaén
Departamento de Informática

Prácticas de Estructuras de Datos
Grado en Ingeniería en Informática
Curso 2023/2024

Práctica 1.b Introducción a los contenedores: vectores (II)

Sesiones de prácticas: 2

Introducción

En la primera parte de esta práctica se ha hecho una introducción a un vector básico que se adapta fácilmente a diferentes contextos en los que se quiera hacer uso de ellos al ser implementado como una plantilla. En esta segunda parte, introduciremos algunas modificaciones sobre la práctica resultante de la sesión anterior hasta implementar un vector dinámico. Este tipo de contenedores intentan mejorar las funcionalidades de un vector básico 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.

Objetivos

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

Descripción de la EEDD

A partir de la clase `VDinamico<T>` implementada en la sesión anterior, modificar y ampliar el contenido de dicha clase 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:

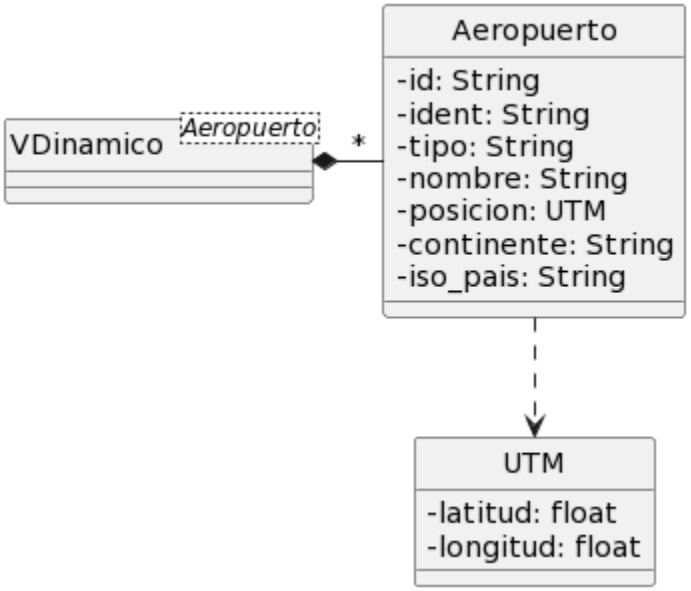
- Constructor por defecto VDinamico<T>(), iniciando el tamaño físico a 1 y el lógico a 0.
- Constructor parametrizado dado un tamaño lógico inicial, VDinamico<T>(unsigned int tamlog), iniciando el tamaño físico a la potencia de 2 inmediatamente superior a tamlog (tamaño lógico).
- Constructor copia VDinamico<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 e incrementa el tamaño físico si fuera necesario.
- 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. Esta operación siempre decrementa el tamaño lógico en 1 y decrementa el tamaño físico si se dan las condiciones.

- 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. Si no, devuelve -1.
- unsigned int tamlog() para obtener el tamaño (lógico) del vector.
- El destructor correspondiente.

Programa de prueba: Gestión de un vector dinámico de Aeropuertos

Con la EEDD vector dinámico implementada, la instanciamos para gestionar una serie de aeropuertos localizados en distintos lugares del mundo de forma que podamos añadir uno nuevo dinámicamente al vector sin preocuparnos del tamaño que este necesite. Para ello, almacenaremos los datos desde el fichero que aparece en Platea junto a este enunciado de la práctica ("aeropuertos.csv") en el nuevo vector dinámico de aeropuertos. El diagrama UML de la práctica es el siguiente:



La clase Aeropuerto tendrá los atributos que aparecen en el UML más los constructores y operadores que necesiteis.

Crear un proyecto con un programa de prueba que realice las operaciones que se detallan a continuación.

- Modificar la plantilla VDinamico<T> para añadir la funcionalidad enumerada anteriormente.
- Leer el fichero "aeropuertos.csv" y almacenar los aeropuertos en un vector dinámico (usar el código adjunto del main.cpp).
- Ordenar el vector de mayor a menor y mostrar los id de los primeros 30 aeropuertos.
- Ordenar el vector de menor a mayor y mostrar los id de los primeros 30 aeropuertos.

- Una vez ordenado el vector, buscar los aeropuertos con los id 345166, 6640, 6676, 345364 y 6778, mostrando su posición en el contenedor y teniendo en cuenta que pueden no existir.

- Eliminar todos los aeropuertos cuyo continente sea “NA”. Para ello, pasar todos los aeropuertos que cumplan la condición a un nuevo vector dinámico y, tras esto, borrar los aeropuertos del vector dinámico original. Mostrar el tamaño lógico de ambos vectores y toda la información de los primeros 20 aeropuertos del nuevo vector. Si se usan vectores auxiliares, deberán declararse como VDinamico<T>.
- 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.