Universidad de Jaén

Departamento de Informática

Prácticas de Estructuras de Datos

Grado en Ingeniería en Informática

Curso 2019/2020

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

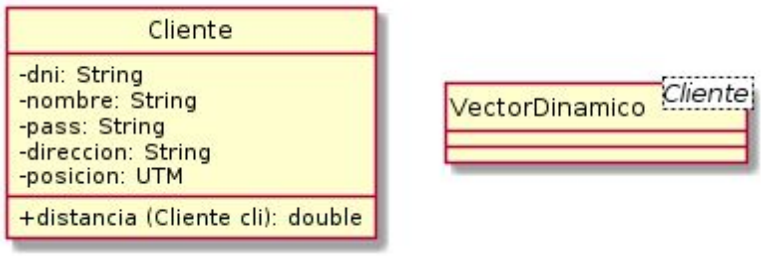
Implementar la clase `VDinamico<T>` para que tenga toda la funcionalidad del vector dinámico descrita en la Lección 4, utilizando patrones de clase y excepciones. Los métodos a implementar serán los siguientes:

- 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 tam)`, iniciando el tamaño físico a la potencia de 2 inmediatamente superior a tam (tamaño lógico).
- Constructor copia `VDinamico<T>(const VDinamico<T>& origen)`.
- Constructor de copia parcial `VDinamico<T>(const VDinamico<T>& origen, unsigned int inicio, unsigned int num)`.
- 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.
- 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 clientes atendiendo al nombre. Se puede utilizar la función sort de <algorithm>: `void ordenar()`.
- 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 tam()` para obtener el tamaño (lógico) del vector.
- El destructor correspondiente.

Programa de prueba: Creación de un vector dinámico de clientes

En esta primera práctica almacenaremos los datos del fichero adjunto en un vector dinámico, es decir, el vector se instanciará con los datos de tipo Cliente. Para ello se deberá leer el fichero adjunto que almacena un cliente por línea. El código para la lectura de dicho fichero se encuentra adjunto a la práctica.



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

- Instanciar el vector con todos los objetos de tipo Cliente leídos desde el fichero en formato csv proporcionado.
- Crear otro vector a partir de éste con los nombres ordenados. Para que la ordenación pueda realizarse hay que sobrecargar el *operator*< en Cliente .
- Eliminar de éste último los clientes que se llamen con un determinado nombre, por ejemplo “Francesco”. Realizar previamente la búsqueda de forma eficiente.
- Calcular la distancia entre los clientes más alejados utilizando directamente los valores de sus coordenadas UTM (UTM es un struct con dos campos, latitud y longitud).

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.