Universidad de Jaén
Departamento de Informática

Prácticas de Estructuras de Datos
Grado en Ingeniería en Informática
Curso 2017/2018

Práctica 2. Implementación de vector dinámico mediante plantillas y operadores en C++

Sesiones de prácticas: 2

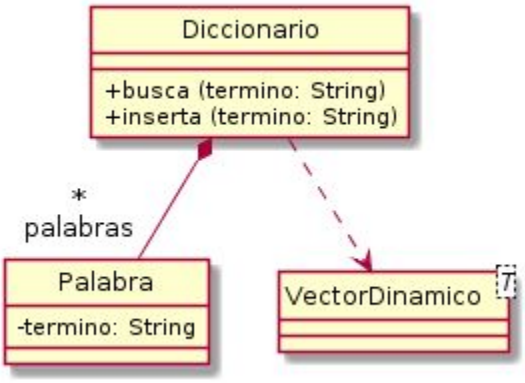
Objetivos

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 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 inicio, unsigned 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 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 pos = UINT_MAX)`. Si no se indica la posición (valor `UINT_MAX`) entonces se elimina el último dato del vector.
- `unsigned tam()` para obtener el tamaño (lógico) del vector.
- El destructor correspondiente.



Programa de prueba: Creación de un diccionario y agregación de nuevas palabras

En esta práctica se ampliará la funcionalidad del Diccionario dotándolo de un método para añadir nuevas palabras (`void insertar(std::string& termino)`). Para garantizar que la eficiencia de la búsqueda en el diccionario se mantiene, el elemento a insertar se colocará de forma ordenada en el vector, descartando, en su caso, los elementos duplicados[1].

En primer lugar, el vector estático introducido en la práctica anterior será cambiado por un vector dinámico instanciado a la clase Palabra (`VDinamico<Palabra>`). Para comprobar su funcionamiento se creará un objeto Diccionario usando el constructor de la práctica anterior `Diccionario(const std::string& ruta)`, el cual inicializará el vector a vacío e irá añadiendo una a una las palabras del fichero proporcionado (por ahora seguimos suponiendo que las palabras del fichero están ordenadas por lo que simplemente se irán añadiendo a la derecha del vector). A continuación se creará un programa de prueba que nos permita introducir frases en español. Cada una de las palabras que no estén introducidas en el diccionario (como conjugaciones verbales, o plurales) se introducirán en él.

Lectura de palabras desde teclado

Para obtener una a una las palabras del texto introducido por el usuario, podemos hacer uso del operador >> que, como sabemos, obtiene una secuencia de caracteres de un flujo de entrada hasta que encuentre caracteres de separación de palabra (espacio, tabulador, fin de línea, retorno de carro…) que serán descartados[2]. Como *std::cin* es un flujo de entrada, podemos utilizar los mismos métodos y funciones de manejo de flujos que utilizabamos para la lectura de un fichero[3].

---

[1] Para la inserción ordenada considere antes qué opción sería más eficiente: colocar el nuevo elemento a la derecha y reordenar el vector completo o localizar la posición de inserción y desplazar los elementos a la derecha antes de asignar el nuevo elemento. ¿Qué eficiencia se obtendría en cada caso?

[2] Ojo, los caracteres de puntuación o símbolos especiales también serían considerados parte de una palabra, por lo que procuraremos no introducirlos en el texto. Para más información consultar http://www.cplusplus.com/reference/istream/istream/operator%3E%3E/

[3] El final de fichero en cin se indica con el carácter Ctrl+d

```
std::cout <<"Introducir frases y pulsar enter "
              <<"(Ctrl+D para finalizar)"<< std::endl;
while (!std::cin.eof())  {
      std::cin >> palabra;
      if (palabra!="") {
            std::cout << ++total << " " << palabra << std::endl;
            palabra="";
      }
}
std::cout << "TOTAL PALABRAS: "<< total << std::endl;
```

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.