PRÁCTICA 2 Estructuras de datos y algoritmos

Título: Programación de TAD genérico arborescente en C++ Curso 2019/20

Objetivo:

- Diseñar una implementación arborescente y en memoria dinámica en C++ del TAD genérico "*colección con repetidos ordenada*", y usarlo para implementar un programa que gestione una colección de libros de una biblioteca, identificados por un código, pero con la posibilidad de que haya varios ejemplares indistinguibles de cada libro (compartiendo código).

Fecha límite de entrega: 19 de diciembre de 2019 (incluido)

Descripción detallada.

Se trata de replicar lo realizado en la Práctica 1, pero desarrollando una **implementación dinámica utilizando un árbol binario de búsqueda** del TAD genérico "colección con repetidos ordenada". Al igual que en la Práctica 1, el iterador deberá permitir recorrer todos los elementos de la colección **por valores crecientes de la clave**.

Observaciones.

- **El código fuente entregado será compilado y probado en *hendrix,* que es donde deberá compilarse y funcionar correctamente.**
- **El código fuente entregado deberá poderse compilar correctamente con la opción –std=c++11 activada.**
  - Esto significa que si se trabaja con la línea de comandos, deberá compilarse con:
    `g++ -std=c++11 ficheros_compilar...`
  - Si se trabaja con CodeBlocks, el proyecto de la práctica deberá estar configurado con la opción `"Have g++ follow the C++11 ISO C++ language standard [-std=c++11]"` activada (localizable en los menús de CodeBlocks, seleccionando sucesivamente: `"Settings"->"Compiler"->->"Global compiler settings"-> pestaña "Compiler Flags"`)
  - Si se trabaja con CodeLite, el proyecto de la práctica deberá estar configurado con la opción `"Enable C++11 features [-std=c++11]"` activada (localizable haciendo clic sobre la carpeta del proyecto en CodeLite y seleccionando sucesivamente: `"Settings.."->"Compiler"->"C++ compiler options"`)

- Todos los ficheros con código fuente que se presenten como solución de esta práctica deberán estar correctamente documentados.
- En el comentario inicial de cada fichero de código fuente se añadirán los nombres y NIA de los autores de la práctica.
- **Los TAD deberán implementarse siguiendo las instrucciones dadas en las clases y prácticas de la asignatura, y no se permite utilizar Programación Orientada a Objetos.**
- No se permite usar las clases o componentes de la *Standard Template Library (STL)*, ni similares.
- **Todas las indicaciones que se dan en los enunciados de las prácticas respecto a nombres de ficheros, programas, opciones del programa, formatos de los ficheros de entrada o de los ficheros de salida que deban generarse, etc., deben cumplirse escrupulosamente para que la práctica sea evaluada.**
- El código fuente del programa de prueba (*main*) deberá encontrarse en un fichero llamado *"practica2.cpp"*, y cumplir escrupulosamente con el funcionamiento y formatos que se han descrito en el enunciado.
- La ruta del fichero de entrada deberá ser "entrada.txt" (no "entrada1.txt", "mientrada.txt", "datos/entrada.txt", ni similares). Ídem para el fichero "salida.txt".
- **La salida debe seguir las especificaciones del enunciado de la Práctica 0 (y por tanto de la Práctica 1).** Por ejemplo, cuando escribimos "insercion CANCELADA: " en el fichero de salida, está escrito sin tilde, la primera palabra en minúsculas, la segunda palabra en mayúsculas, y con un espacio en blanco entre las dos palabras, y otro espacio en blanco tras el carácter ':'. De forma similar, será obligatorio cumplir todos los formatos descritos en el enunciado.

Material a entregar. Instrucciones.

- **La práctica sólo deberá someterla <u>uno</u> de los miembros del equipo de prácticas desde su cuenta de *hendrix***, y preferiblemente siempre el mismo para todas las prácticas.
- Conectarse a hendrix-ssh.cps.unizar.es según se explica en el documento "Realización y entrega de prácticas en los laboratorios del DIIS" disponible en moodle.
- Crear un directorio, llamado B_p2 si tu profesor tutor es Jorge Bernad, C_p2 si tu profesor tutor es Javier Campos, o V_p2 si tu profesora tutora es Yolanda Villate, donde se guardará un directorio *practica2* que contenga todos los ficheros desarrollados para resolver la práctica (este directorio, *practica2*, deberá contener todos los ficheros con código fuente C++ necesarios para resolver la práctica y dos <u>ficheros de texto, ***entrada.txt*** y ***salida.txt*****,**</u> con los formatos explicados pero <u>que sean significativamente diferentes</u> a los proporcionados como ejemplo, y con los que habréis probado la implementación realizada en vuestra práctica). A la hora de evaluar la práctica se utilizará tanto el fichero de prueba que se entregue, como ficheros de prueba entregados por otros compañeros, o ficheros propios de los profesores.

- Crear el fichero X_p2.tar, con X igual a B, C o V, dependiendo de quién sea tu profesor tutor, con el contenido del directorio X_p2 ejecutando el comando

```
tar –cvf X_p2.tar X_p2
```

- Enviar el fichero X _p2.tar, con X igual a B, C o V, dependiendo de quién sea tu profesor tutor, mediante la orden

```
someter –v eda_19 X_p2.tar
```

ADVERTENCIA: la orden `someter` no permite someter un fichero si el mismo usuario ha sometido antes otro fichero con el mismo nombre y para la misma asignatura, por lo tanto, **antes de someter vuestra práctica, aseguraos de que se trata de la versión definitiva que queréis presentar para su evaluación.**